Trabajo Fin de Grado
Ingeniería de las Tecnologías Industriales

Modelado, control y simulación de manipuladores aéreos incluyendo modelos de sensores

Autor: Manuel Orta Hernández
Tutor: Aníbal Ollero Baturone

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado
Ingeniería de las Tecnologías Industriales

Modelado, control y simulación de manipuladores aéreos incluyendo modelos de sensores

Autor:
Manuel Orta Hernández

Tutor:
Aníbal Ollero Baturone
Catedrático de Universidad

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Trabajo Fin de Grado: Modelado, control y simulación de manipuladores aéreos incluyendo modelos de sensores

Autor: Manuel Orta Hernández

Tutor: Aníbal Ollero Baturone

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

Agradecimientos

A mi familia, que ha creído en mí sin un atisbo de duda desde el momento en que tuve ganas de estudiar esta carrera. Que me han estado manteniendo durante ya más de cuatro años no sólo económicamente, sino también emocionalmente. El calor que he recibido de ellos ha sido la fuente de energía que me ha permitido pasar días y días estudiando, noches en vela y volver a intentarlo tras cada suspenso hasta conseguirlo.

A Irene, por su increíble capacidad para motivarme, organizarme y en definitiva hacerme mejor persona.

A Álvaro Caballero por su dedicación en la guía de este trabajo, implicándose mucho en él y ayudándome a sacarlo adelante pese a mis malas decisiones.

Resumen

Los manipuladores aéreos son vehículos aéreos no tripulados dotados de capacidad de manipulación gracias al acoplamiento de uno o varios brazos robóticos u otros efectores finales sobre una plataforma aérea. En este trabajo se realiza el estudio completo sobre estos sistemas, pasando por el modelado dinámico, el control y la simulación de un elemento de este tipo, con configuración de quadrotor y brazo manipulador planar de dos grados de libertad rotativos.

Primero se diseña el modelo y controlador de la plataforma aérea, después se hace lo mismo para el manipulador y por último se obtiene el modelo de plataforma y brazo acoplados y se utilizan los controladores ya obtenidos para su control. Todos los modelos y simulaciones han sido desarrollados en Simulink y la verificación de todos los resultados se ha realizado por medio de simulaciones.

Además, se aborda el modelado, implementación y simulación de los sistemas de sensorización del elemento, habiéndose creado dos modelos distintos para la adición del ruido de medida y realizándose una comparativa entre ambos. El tratamiento con sensorización pone de manifiesto la necesidad de la existencia de una etapa de estimación de estado, lo cual en nuestro trabajo se ha traducido en la implementación de un filtro de Kalman para ello.

Abstract

Aerial manipulators are unmanned aerial vehicles gifted with manipulation capacity, thanks to the coupling of one or more robotic arms or other end-effectors to an aerial platform. In this work, a complete study about these systems is made, attending to dynamic modelling, control and simulation of an aerial manipulador with quadrotor configuration and a robotic arm of two degrees of freedom and flat motion.

First, the model and controller of the platform is designed, then it is made the same for the manipulator and finally the model of the coupled platform and arm are obtained and controlled by the previously designed controllers. All the models and simulations are developed in Simulink, and the verification of all the results is made by means of simulations.

Furthermore, the modelling, implementation and simulation of the sensors systems is addressed, and there were created and comparated two different models for the noise addition. The treatment of the sensors forces the inclusion of an state estimation stage, and it is done in this work trough the implementation of a Kalman Filter.

Índice

Índice de Tablas

Índice de Figuras

Notación

A^T	Traspuesta
F, v	(En negrita) Vector
u × **v**	Producto vectorial
I	Matriz identidad
sin(α)	Seno del ángulo
cos(α)	Coseno del ángulo
S_i	Seno de q_i
C_i	Coseno de q_i
q_i	Posición generalizada
u_i	Velocidad generalizada

1 INTRODUCCIÓN

La configuración basada en rotores de algunos vehículos aéreos como los helicópteros o los drones multirrotor, les dota de algunas capacidades de vuelo muy particulares, tales como la capacidad de vuelo en punto fijo (*hovering*), despegue y aterrizaje vertical (*Vertical take-off and landing, VTOL*) y ciertas piruetas que no pueden conseguirse con configuraciones clásicas como las de ala fija. Los vehículos aéreos no tripulados (*Unmanned Aerial Vehicles, UAV*) con este tipo de construcción están siendo estudiados extensivamente en la actualidad, dado que esas capacidades de vuelo son especialmente interesantes para una gran cantidad de aplicaciones como pueden ser la vigilancia, filmación aérea o tomas precisas de medidas y datos [2].

Pero esas capacidades tan especiales de vuelo no sólo pueden ser aprovechadas para aplicaciones pasivas como la fotografía o la toma de medidas. Existen múltiples proyectos en los cuales se trata de dotar a una plataforma aérea de capacidad de manipulación. Tal y como se dice en [4], "La manipulación aérea trata sobre robots aéreos equipados con brazos robóticos u otros efectores finales, que realizan tareas tales como montaje o inspección con contacto en lugares inaccesibles, o peligrosos y caros de acceder desde el suelo. Involucra problemas muy desafiantes, principalmente debidos a las rápidas dinámicas con las que se trabaja comparado con otros sistemas de manipulación con bases flotantes."

Figura 1.1 UAV de Aeroarms realizando tareas de manipulación.

Fuente: [9]

Estos proyectos son resultado del interés por parte de los órganos de investigación para desarrollar proyectos de manipulación aérea. Un ejemplo de esto es el proyecto Aeroarms [9], el cual es llevado a cabo por diez *partners* liderados por la Universidad de Sevilla y que está financiado dentro del Programa Marco de Investigación e innovación de la Unión Europea: Horizon 2020. Aeroarms ha conseguido demostrar la capacidad y adecuación de este tipo de sistemas aéreos para la inspección con contacto en entornos industriales.

Figura 1.2 UAV de Aeroarms realizando inspección de contacto en tubería industrial.

Fuente: [9]

La mejora que estos sistemas pueden aportar en ciertos trabajos con respecto a la forma convencional de realizarlos les convierte en un excelente producto si se enfoca al cliente adecuado. Por esta razón, ya existen empresas privadas que invierten en investigación y desarrollan en este campo. Por ejemplo, con el eslogan "Revolutionary Drones for Professionals", la empresa japonesa Prodrone ® ya oferta sistemas manipuladores aéreos completos (además de otros sistemas *clásicos* de filmación o de transporte de cargas).

Figura 1.3 UAV manipulador de Prodrone.

Fuente: [10]

El Centro Aeroespacial Alemán (DLR) también se involucra en el estudio de la manipulación aérea. En la siguiente imagen, que ellos mismos titulan "*Autonomous double-rotor helicopter with mounted robotic manipulator during flight experiment*" en su página web, podemos ver un manipulador aéreo de tipo pequeño helicóptero con altas capacidades de carga.

Figura 1.4 Manipulador aéreo de DLR

Fuente: [11]

El siguiente paso en este hilo de investigación es la coordinación de dos o más UAV en sus tareas, tanto de vuelo como de manipulación. De esta manera se pueden realizar trabajos aún más complejos, si cabe, gracias a la cooperación de varios robots. Multitud de universidades estudian actualmente el control y planificación de múltiples robots autónomos; destacan, por ejemplo, los artículos de la Universidad de Seúl:

Figura 1.5 Cooperación de dos drones en el transporte de cargas

Fuente: [5]

En la Figura 1.5 podemos observar el resultado de un estudio en el que se trata de coordinar dos multirrotores para que colaboren en el transporte de una carga, con la dificultad añadida de que existe un obstáculo entre los puntos inicial y final, por lo que los UAV han de planificar una nueva trayectoria para esquivarlo. También encontramos estudios publicados por la misma universidad, referentes a la coordinación de múltiples vehículos como [6].

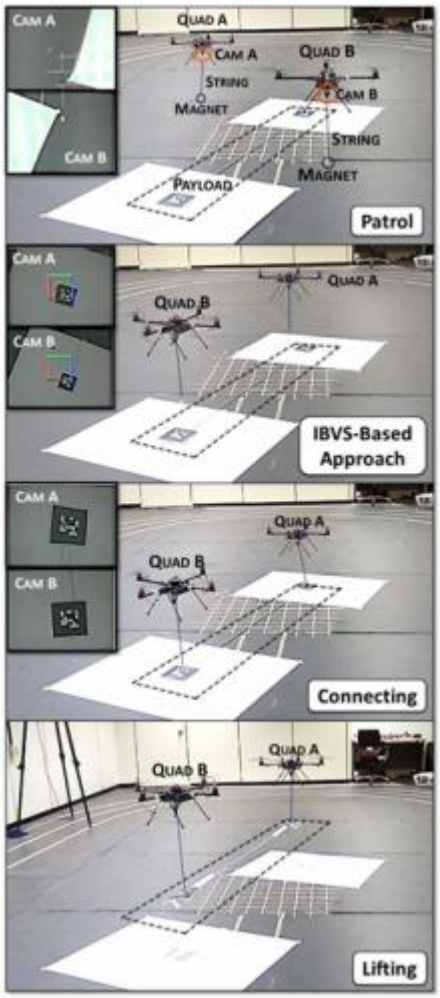


Figura 1.6 Colaboración en levantamiento de carga basada en visión

Fuente: [6]

Otras universidades que estudian la coordinación de varios vehículos aéreos en tareas de manipulación son, por ejemplo, la Universidad de Palermo, con artículos también en el campo de la evitación de obstáculos [18]. También la Universidad de Sevilla ha publicado artículos referentes a la coordinación de múltiples UAVs [19].

1.1 En vista a este trabajo

En este proyecto se va a realizar paso a paso el modelado y control de un robot aéreo autónomo dotado de un brazo manipulador (en definitiva, un UAV manipulador como los que hemos estado nombrando hasta ahora). Como comentamos en la introducción, la configuración motriz basada en rotores proporciona una serie de capacidades de vuelo que son especialmente interesantes para las tareas de manipulación, como el VTOL o la capacidad de *hovering*. Las dos configuraciones más estudiadas y extendidas con respecto a este tipo de vehículos aéreos son las de helicóptero y la de multirrotor. En concreto, el tipo de estructura que vamos a elegir para el vehículo será la configuración de multirotor con cuatro motores, y un brazo manipulador simple con dos grados de libertad, formado por dos eslabones con forma de barra rígida y dos vínculos rotativos. El brazo manipulador estará unido a la plataforma mediante la articulación rotativa correspondiente al primer grado de libertad en el centro de la parte inferior, Figura 1.7. No obstante, es importante mencionar que todos los modelos y desarrollos presentados son fácilmente extendibles a configuraciones con más rotores y manipuladores con más grados de libertad.

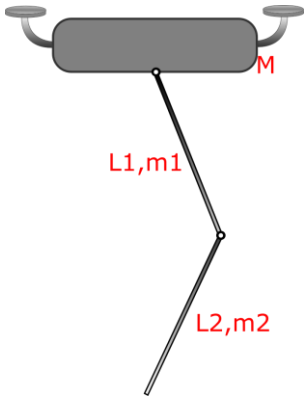


Figura 1.7 UAV + manipulador

Fuente: elaboración propia

La elección de una plataforma tipo quadrotor para el UAV se justifica en las ventajas que esta ofrece frente al tipo helicóptero. La primera y más importante es que la estrategia de control de un multirrotor es más simple que la de un helicóptero. Por ejemplo, los efectos inerciales del rotor principal de un helicóptero que provocan una deriva en el *yaw*[1] no existen en un quadrotor, ya que los cuatro motores giran en sentidos opuestos dos a dos, compensándose los pares producidos como puede verse en la Figura 1.8.

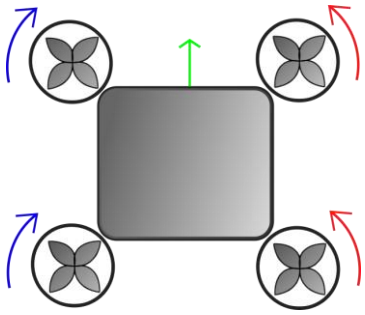


Figura 1.8 Sentido de giro de las hélices en un quadrotor

Fuente: Elaboración propia

Además, una estructura de tipo helicóptero, con la necesidad de un rotor principal más potente y unas palas más grandes es más cara de construir, así como más peligrosa que una estructura tipo multirrotor con características de vuelo similares.

Por el contrario, la capacidad de carga de un helicóptero es sensiblemente mayor que la que puede aportar un multirrotor. Aun así, teniendo en cuenta el balance de ventajas y desventajas, se puede ceder en esta característica ya que los brazos manipuladores que pretendemos equipar en los UAV estudiados en este trabajo son ligeros; así como también serán de baja complejidad las tareas que se realicen con ellos, es decir, no serán tareas de transporte o levantamiento de cargas pesadas, sino pequeños trabajos de manipulación o inspección.

A continuación, definimos cuatro términos clave que aparecerán continuamente en el desarrollo del trabajo, y que conviene aclarar su significado para no llevar a confusión.

UAV	*Unmanned Aerial Vehicles*, Vehículo Aéreo no Tripulado
VTOL	*Vertical Take-off and Landing,* Despegue y aterrizaje vertical
Quadrotor	Llamamos quadrotor a aquellos robots aéreos de tipo VTOL que cuentan con una estructura de tipo multirrotor con cuatro rotores.
Dron	Este término se refería inicialmente a lo mismo que UAV, pero con el tiempo y el gran crecimiento que han sufrido la investigación y el mercado de los multirrotores, el uso de este término cada vez se va sesgando más hacia estos últimos.

[1] Uno de los tres ángulos de navegación (*Roll, Pitch & Yaw* o respectivamente alabeo, cabeceo y guiñada), que corresponde al giro alrededor del eje vertical perpendicular al UAV.

2 MODELO Y CONTROL TRIDIMENSIONAL DE LA PLATAFORMA MULTIRROTOR

En este apartado se obtendrá un modelo de un UAV multirrotor que servirá como plataforma base en la que posteriormente acoplaremos el brazo manipulador. Partiremos de las ecuaciones cinemáticas y dinámicas del sistema presentadas en [1], y desarrollaremos el problema del control hasta poder realizar simulaciones en las que comprobemos el funcionamiento de nuestro multirrotor en condiciones ideales; tampoco será incluido en este primer modelo el comportamiento real de los sensores (este punto será tratado en profundidad más adelante en el trabajo).

La integridad de este trabajo y por tanto también los contenidos desarrollados en esta sección han sido implementados con las herramientas de programación y simulación Matlab y Simulink.

2.1 Modelo

Nota: todos los sistemas de referencia que utilicemos en los desarrollos mecánicos de los sistemas de este trabajo serán cartesianos, ortonormales y dextrógiros.

De acuerdo con [1], podemos dividir el sistema en dos submodelos: el modelo mecánico y el modelo aerodinámico. Este último sería el resultado de complicadas ecuaciones de mecánica de fluidos, pero que para pequeños VTOLs de menos de 20 Kg pueden ser expresadas por relaciones algebraicas relativamente sencillas.

El esquema que representa el sistema tridimensional que queremos modelar es el siguiente:

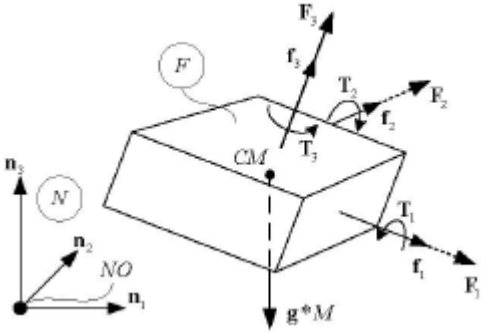


Figura 2.1 Esquema de un UAV genérico

Fuente: [1]

En el que podemos observar:

- El sistema de referencia {n}, con origen fijo en un punto aleatorio en el suelo, y cuya dirección $\mathbf{n_3}$ (vector unitario) corresponde a la coordenada vertical. A lo largo de esta dirección, pero en sentido contrario, aparece la aceleración de la gravedad con valor conocido g.
- El cuerpo F, de masa M, modela un sólido rígido cualquiera (en este caso el quadrrotor). Acoplado a él, y con el origen situado en el centro de masas del sólido (CM) tenemos el sistema de referencia {f}, cuyos vectores unitarios son $\{\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}\}$. Las fuerzas y pares aplicados al sólido en cada uno de los ejes $\mathbf{f_i}$, se denotan como $\mathbf{F_i}$ y $\mathbf{T_i}$.

2.1.1 Cinemática

2.1.1.1 Notación

Introducimos seis coordenadas generalizadas q_i y seis velocidades generalizadas u_i, de tal forma que:

- Llamaremos q_1, q_2 y q_3 a las coordenadas correspondientes a la posición del centro de masas del cuerpo F (CM) en el sistema de coordenadas {n}.
- Llamaremos u_1, u_2 y u_3 a las velocidades correspondientes a q_1, q_2 y q_3, respectivamente. También en el sistema de coordenadas {n}.
- Llamaremos q_4, q_5 y q_6 a los ángulos de Euler, que representan el giro del cuerpo F respecto a N. Son correspondientes a rotaciones sucesivas según los ejes {f} en orden 1, 2 y 3.
- Llamaremos u_4, u_5 y u_6 a las velocidades angulares que derivan de cada uno de los ángulos q_4, q_5 y q_6. En base a esto, podemos decir que:

$$\boldsymbol{\omega_{F-N}} = u_4\boldsymbol{f}_1 + u_5\boldsymbol{f}_2 + u_6\boldsymbol{f}_3 \qquad \textit{Ecuación 2.1}$$

2.1.1.2 Ecuaciones de la cinemática

Dado que u_1, u_2 y u_3 son directamente las celeridades de las posiciones q_1, q_2 y q_3, podemos obtener las ecuaciones para la cinemática de traslación:

$$\dot{q_1} = u_1 \qquad \textit{Ecuación 2.2}$$

$$\dot{q_2} = u_2 \qquad \textit{Ecuación 2.3}$$

$$\dot{q_3} = u_3 \qquad \textit{Ecuación 2.4}$$

Para la cinemática de rotación, tenemos las ecuaciones

$$\dot{q_4} = (u_4\cos(q_6) - u_5\sin(q_6))/\cos(q_5) \qquad \textit{Ecuación 2.5}$$

$$\dot{q_5} = u_4\sin(q_6) + u_5\cos(q_6) \qquad \textit{Ecuación 2.6}$$

$$\dot{q_6} = u_6 + \tan(q_5)(u_4\cos(q_6) - u_5\sin(q_6)) \qquad \textit{Ecuación 2.7}$$

2.1.2 Dinámica

Las ecuaciones de la dinámica traslacional surgen de aplicar el teorema de la cantidad de movimiento, o la segunda ley de Newton:

$$\begin{pmatrix}\dot{u_1}\\ \dot{u_2}\\ \dot{u_3}\end{pmatrix} M = \boldsymbol{C_{F-N}}\begin{pmatrix}F_1\\ F_2\\ F_3\end{pmatrix} + \begin{pmatrix}0\\ 0\\ -g*M\end{pmatrix} \qquad \textit{Ecuación 2.8}$$

Donde $\mathbf{C_{F-N}}$ es la matriz de orientación de F con respecto a N, g es el valor de la aceleración de la gravedad y M es la masa del sistema.

Se va a realizar una importante simplificación cinemática para la rotación que afecta al modelo y por lo tanto su posterior control. Si conseguimos que la dinámica de control de la coordenada generalizada q_6 (*yaw)* sea bastante más rápida que la de todas las demás variables generalizadas (aproximadamente de un orden de magnitud o 10 veces más rápida), podríamos asumir la siguiente igualdad:

$$u_6 = cte$$ ***Ecuación 2.9***

De acuerdo con [1], esto puede ser logrado gracias a un esquema de control con doble lazo, en el que se trate a u_6 de forma separada al resto de variables.

Las tres ecuaciones que nos faltan para tener el sistema identificado son las correspondientes a la dinámica de rotación. Siguiendo el desarrollo de [1], debido a la compensación que ocurre al tener dos motores girando en sentido positivo y dos en negativo, la dinámica de giro del quadrotor viene dada principalmente por la inercia del cuerpo. Este comportamiento está bien aproximado por las ecuaciones de Euler:

$$T_1 + (I_{33} - I_{22})u_5u_6 - I_{11}\dot{u}_4 = 0$$ ***Ecuación 2.10***

$$T_2 - (I_{33} - I_{11})u_4u_6 - I_{22}\dot{u}_5 = 0$$ ***Ecuación 2.11***

$$T_3 + (I_{22} - I_{11})u_4u_5 - I_{33}\dot{u}_6 = 0$$ ***Ecuación 2.12***

Donde $I_{ii}, i = 1, 2, 3$ son los valores de las inercias respecto al centro de masa en los ejes $\boldsymbol{f_{1,2,3}}$.

A la vista de las ecuaciones anteriores para la dinámica de rotación, comprobamos que la asunción de que u_6 es constante desde el punto de vista del resto del sistema (Ecuación 2.9) gracias al control en un lazo separado simplifica mucho el sistema, convirtiendo la dinámica rotacional en lineal.

2.1.3 Implementación

La integración en Simulink de todas estas ecuaciones que modelan la cinemática y dinámica, las escribiremos en un bloque "Matlab Fcn", que permite escribir directamente las ecuaciones que queremos aplicar para una serie de variables de entrada y de salida e implementaremos en nuestro modelo para simulación de la siguiente forma:

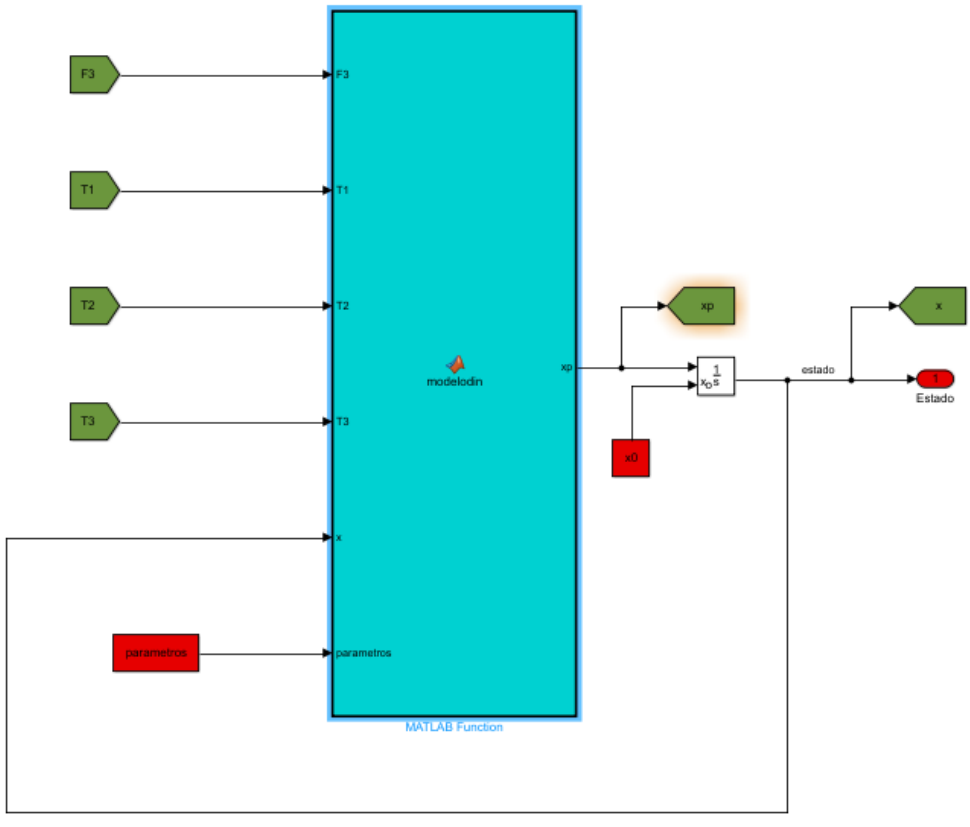


Figura 2.2 Implementación del modelo mecánico del UAV

Fuente: elaboración propia

Como se observa en la figura, las entradas de actuación son las variables dinámicas F_3, T_1, T_2 y T_3 (esta última es el par resultante del control del *yaw*, que veremos más adelante). Estas variables dinámicas actúan sobre el sistema mecánico ocasionando un movimiento, que es la salida del sistema, y estructuraremos en un vector de estado de doce componentes.

$$x = [q_1, q_2, q_3, q_4, q_5, q_6, u_1, u_2, u_3, u_4, u_5, u_6]^t$$

También se ven en la figura el vector x_0, que contiene las condiciones iniciales de cada variable con la misma estructura que x, y el vector "parámetros", que ahora contiene las constantes I_{11}, I_{22} e I_{33} además de M y g.

Cabe mencionar que la salida del modelo mecánico es el vector de estado derivado, $\dot{x}$, pero obtenemos x con un integrador, cuyas condiciones iniciales son el vector antes mencionado x_0.

El código dentro del bloque "Matlab Fcn" de Simulink que modela al sistema son todas las ecuaciones de la cinemática y dinámica que se han visto en el apartado "2.1 Modelo" de esta memoria, y podemos verlo a continuación.

```
function xp = modelodin(F3,T1,T2,T3,x,parametros)

%ENTRADAS

q1=x(1); q2=x(2); q3=x(3); q4=x(4); q5=x(5); q6=x(6);
u1=x(7); u2=x(8); u3=x(9); u4=x(10); u5=x(11); u6=x(12);

M=parametros(1);
g=parametros(2);
I11=parametros(3);
I22=parametros(4);
I33=parametros(5);

%CONDICIONES

F1=0;   F2=0;

Cfn=[cos(q5)*cos(q6),                                -cos(q5)*sin(q6),
sin(q5)          ;...
     cos(q4)*sin(q6)+sin(q4)*sin(q5)*cos(q6),     cos(q4)*cos(q6)-
sin(q4)*sin(q5)*sin(q6), -sin(q4)*cos(q5);...
     sin(q4)*sin(q6)-cos(q4)*sin(q5)*cos(q6),
sin(q4)*cos(q6)+cos(q4)*sin(q5)*sin(q6), cos(q4)*cos(q5) ];

%ECUACIONES CINEMÁTICAS

q1p=u1;
q2p=u2;
q3p=u3;
q4p=(u4*cos(q6)-u5*sin(q6))/cos(q5);
q5p=u4*sin(q6)+u5*cos(q6);
q6p=u6+tan(q5)*(u4*cos(q6)-u5*sin(q6));

%ECUACIONES DINÁMICAS

% ([u1p u2p u3p]')*M=Cfn*[F1 F2 F3]'+[0 0 -g*M]';
up=(1/M)*(Cfn*[F1,F2,F3]'+[0,0,-g*M]');

u1p=up(1);
u2p=up(2);
u3p=up(3);
u4p=(T1 + (I33-I22)*u5*u6)/I11;
u5p=(T2 - (I33-I11)*u4*u6)/I22;
u6p=(T3 + (I22-I11)*u4*u5)/I33;
```

```
%SALIDAS

xp=[q1p,q2p,q3p,q4p,q5p,q6p,u1p,u2p,u3p,u4p,u5p,u6p];
```

2.2 Control

Los vehículos con VTOL como los helicópteros o los multirrotores que estamos considerando en este trabajo tienen una forma particular de moverse, muy distinta de otras configuraciones como la de ala fija. La forma de actuar sobre el sistema es ajustando la orientación del robot y el módulo de la fuerza vertical $\mathbf{F_3}$ (Figura 2.1). Por ejemplo, para mover el dron hacia delante, una vez que esté suspendido en el aire tendríamos que inclinarlo de forma que los dos rotores traseros quedaran más elevados que los dos delanteros. Esta inclinación significa un cambio en la dirección de la fuerza vertical $\mathbf{F_3}$, resultando no nula su proyección en el plano horizontal del sistema de referencia {n}. La dirección hacia la que se desplaza (*yaw)*, se controla en un *loop* separado, como mencionamos en el apartado anterior.

Aplicaremos la estructura clásica de control en bucle cerrado:

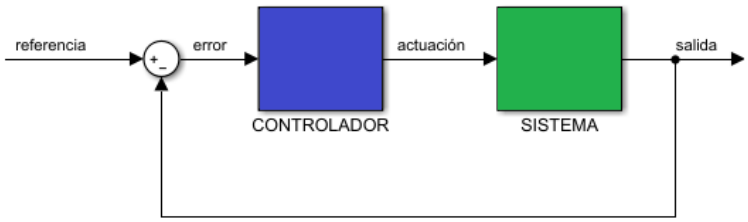


Figura 2.3 Esquema de control realimentado

Fuente: elaboración propia

Estructura de la cual tendremos dos lazos de control separados. El primero será para la posición del robot, cuyas referencias serán las coordenadas de posición q_1, q_2 y q_3. El otro lazo, que controlará la orientación del robot, tendrá como consigna la orientación deseada q_6 (*yaw*).

Basándonos en los estudios realizados en [1], el primer lazo de control, o control de traslación del móvil tendrá a su vez dos bucles. Uno externo o principal en el que el controlador trata de llevar las coordenadas de posición q_1, q_2 y q_3 hacia las referencias dadas para la posición del UAV, y otro interno encargado de controlar la inclinación o rotaciones del dron q_4 y q_5. Este lazo interno es necesario por lo que se comentó al principio de esta sección relativo a la forma en la que se desplazan estos vehículos: las coordenadas q_4 y q_5 codifican los llamados *pitch & roll*, por lo que son ellas las que reflejarán cualquier inclinación de nuestro robot necesaria para su desplazamiento.

2.2.1 Control de traslación y rotaciones longitudinal y lateral

A continuación, trataremos de implementar en Simulink® el controlador para los movimientos de traslación propuesto en [1]. Cualquier movimiento en el plano horizontal del UAV implica unos giros en los ángulos de *pitch & roll*, por lo que el controlador de estos ángulos está incluido en esta sección. A continuación, se realiza un desglose de cada componente del bucle de control.

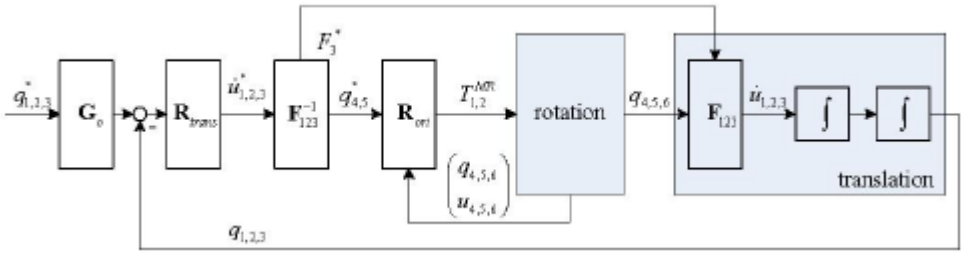


Figura 2.4 Estructura del control de traslación y rotaciones laterales

Fuente: [1]

El primer bloque "G_o" no es más que un prefiltro para las señales de referencia, que en nuestro caso no necesitaremos puesto que, a falta de una fase de planificación o generadora de trayectorias, estableceremos consignas adecuadas para nuestro sistema desde el punto de vista dinámico.

En la siguiente tabla se resume la notación utilizada:

Término	Significado
q	**Posición real**
q*, qd	**Posición deseada**
$\dot{q}$, qp	**Derivada**
$\ddot{q}$, qpp	**Segunda derivada**
$\dot{u}$, up	**Aceleración**
x	**Vector de estado**

Tabla 2-1 Notación utilizada

Nota: Para las velocidades la notación es la misma, cambiando q por u.

El bloque "Rtrans" es el controlador para el lazo externo, es decir, contiene la implementación de los PID[2] para cada una de las tres coordenadas de desplazamiento. La forma de actuar sobre el vehículo para cambiar su posición es a través de las **aceleraciones traslacionales** (ver Tabla 2-1 Notación utilizada), por lo que estas serán las salidas del bloque de control "Rtrans". Dado que las referencias son las coordenadas de posición deseadas, realimentamos el bucle y calculamos el error con las posiciones reales, tomadas del vector de estado. La ley de control resultante, tal y como presenta [2] en su estudio para el control de UAVs atados al suelo es la siguiente:

$$\dot{u_i}^* = K_p(q_i^* - q_i) + K_I \int (q_i^* - qi)dt - K_D u_i \qquad \textit{Ecuación 2.13}$$

i = 1, 2, 3

Definimos el error:

$$e_i = q_i^* - q_i \qquad \textit{Ecuación 2.14}$$

La forma de implementar la ley de control (Ecuación 2.13) para cada una de las coordendas es la siguiente:

[2] Controlador de acción Proporcional, Integral y Derivativa

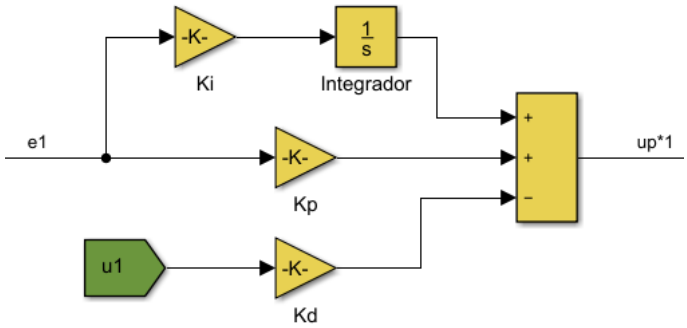


Figura 2.5 Controlador PID en Simulink

Fuente: elaboración propia

Donde e_1 se calcula como se especificó anteriormente (Ecuación 2.14).

A continuación, necesitamos traducir las aceleraciones traslacionales a nuestras variables de acción para el movimiento del UAV (q_4, q_5 y F_3 o lo que es lo mismo, inclinación de la plataforma y fuerza de sustentamiento vertical). Esta tarea es realizada por el bloque "F_{123}^{-1}", cuyas ecuaciones son obtenidas al resolver las ecuaciones de la dinámica de traslación (Ecuación 2.8) para las variables q_4, q_5 y F_3. Para esta resolución es necesaria la asunción siguiente:

$$F_1 = F_2 = 0 \qquad \textit{Ecuación 2.15}$$

La resolución de dichas ecuaciones, y por lo tanto las expresiones a implementar en el bloque F_{123}^{-1} son las siguientes:

$$F_3^* = M\sqrt{\dot{u_1^2} + \dot{u_2^2} + (u_3 + g)^2} \qquad \textit{Ecuación 2.16}$$

$$q_5^* = \arcsin\left(M\frac{u_1}{F_3}\right) \qquad \textit{Ecuación 2.17}$$

$$q_4^* = -\arcsin\left(M\frac{u_2}{F_3\cos\left(q_5\right)}\right) \qquad \textit{Ecuación 2.18}$$

La forma de implementarlos es mediante un bloque "Matlab Fcn" de Simulink.

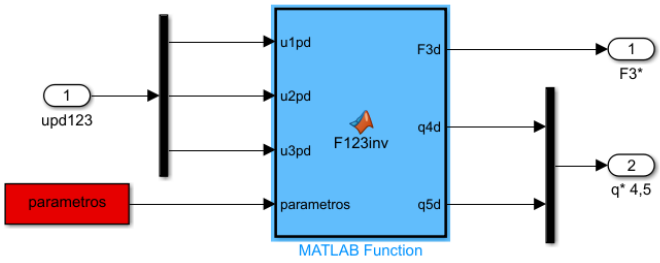


Figura 2.6 Bloque F_{123}^{-1}

Fuente: elaboración propia

Nota: La variable "parámetros" es un vector que contiene los valores de g y M.

En este punto del desarrollo conocemos la orientación y la fuerza vertical deseadas, todo lo necesario para el movimiento de traslación del robot. Eso significa que ahora comenzamos a implementar el lazo interno de control, o lazo de orientación. Aunque este lazo secundario también implica rotaciones, en este caso estamos hablando del *pitch & roll,* los cuales no debemos nunca confundir con el *yaw*, que se controla en otro bucle a parte que aún no hemos detallado.

El lazo interno para la orientación, en el esquema general del control del UAV (Figura 2.4 Estructura del control de traslación) corresponde a los dos bloques "Rori" y "rotation", cuyo esquema completo podemos observar a continuación:

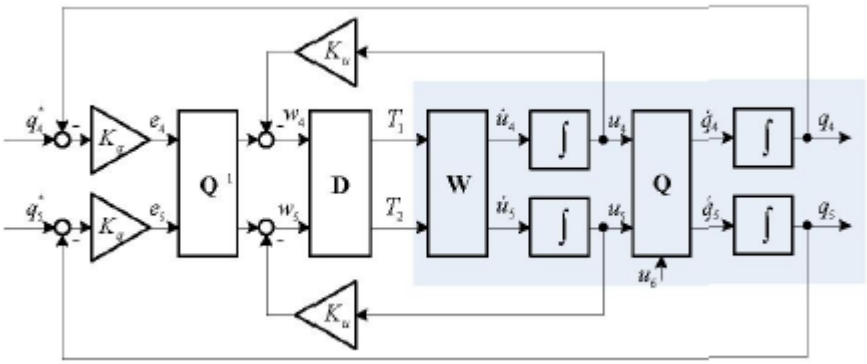


Figura 2.7 Lazo interno para la orientación

Fuente: [1]

De nuevo nos encontramos ante una estructura compuesta por dos lazos de realimentación, en este caso anidados. El primero es el lazo de realimentación de q_4 y q_5, controladas con una ley proporcional de valor K_q. El segundo, que también es un controlador proporcional en este caso de constante K_u, se ocupa de las variables u_4 y u_5.

Si nos fijamos en otro estudio sobre el control de VTOLs, [2], nos encontramos con la estructura siguiente:

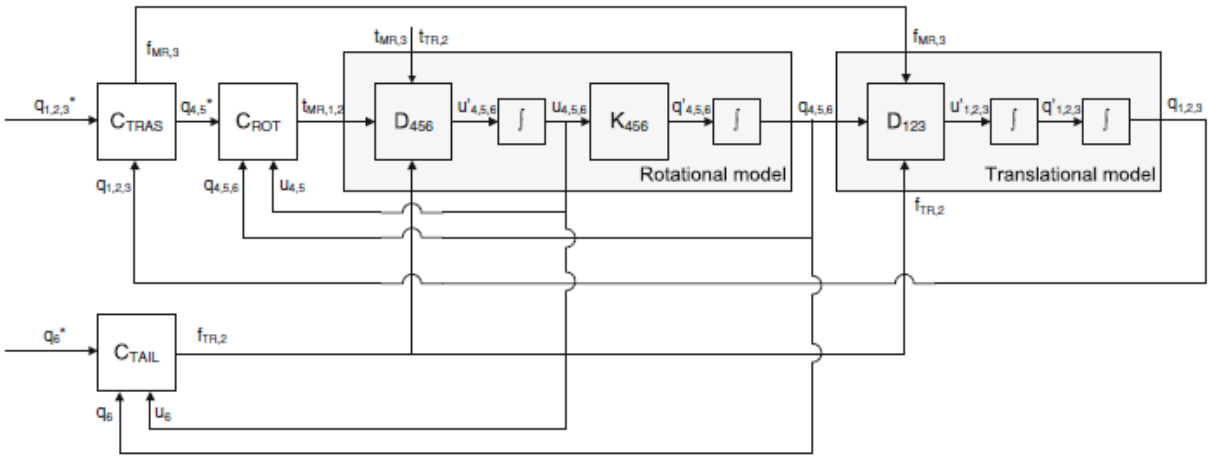


Figura 2.8 Estructura del control de traslación (II)

Fuente: [2]

Pese a controlar un VTOL de tipo helicóptero con otras mejoras añadidas, encontramos la misma estructura de control. Hasta ahora, en nuestro desarrollo, hemos llegado hasta "Crot" que en el desarrollo de [1] equivale al bloque "Rori", pasando por "Ctras" que en [1] llamábamos "Rtrans". Mientras que en [1] se detallaba el interior de Rori en la primera mitad del esquema del control de orientación (Figura 2.7 Lazo interno para la orientación), en [2] podemos ver la estructura del control del lazo interno en la siguiente figura:

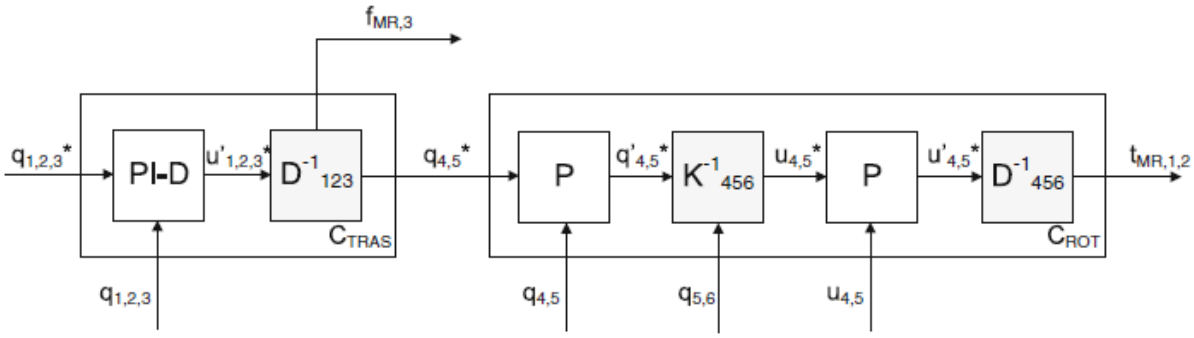


Figura 2.9 Lazo interno para la orientación (II)

Fuente: [2]

Fijándonos en el cuadro de la derecha, para ver el interior de "Crot", observamos que se ha aplicado en cada uno de los controladores proporcionales la ley de control:

$$\dot{q_i^*} = K_p(q_i^* - q_i) \qquad \textit{Ecuación 2.19}$$

i = 4, 5

Efectivamente, en [1] se ha aplicado la misma ley de control para los controladores proporcionales, pero con la notación cambiada en la que $\dot{q_i^*}$ pasa a llamarse e_i, y $\dot{u_i^*}$ se convierte en w_i. En nuestro desarrollo continuaremos utilizando la notación de [2], la cual coincide con lo que establecimos al comienzo (Tabla 2-1 Notación utilizada).

Continuando el recorrido, ahora en el esquema de [2], el bloque K_{456}^{-1} corresponde a las ecuaciones de la cinemática rotacional inversa resueltas para u4 y u5 (Ecuación 2.5 y Ecuación 2.6), cuyas soluciones son:

$$u_4^* = c_5 c_6 \dot{q_4^*} + s_6 \dot{q_5^*} \qquad \textit{Ecuación 2.20}$$

$$u_5^* = -c_5 s_6 \dot{q_4^*} + c_6 \dot{q_5^*} \qquad \textit{Ecuación 2.21}$$

Que implementaremos en el interior del subsistema Q^{-1} o K_{456}^{-1} mediante un bloque "Matlab Fcn" de Simulink.

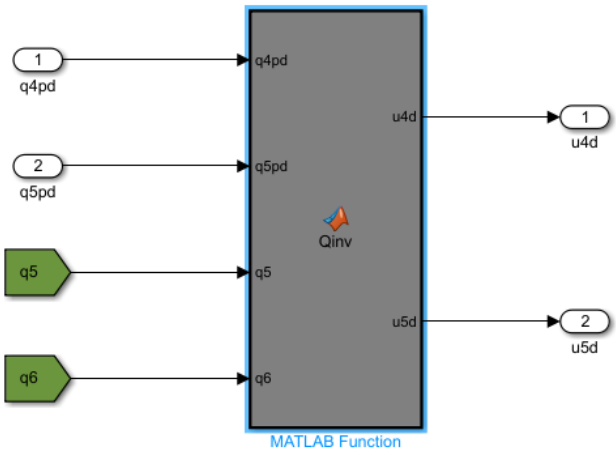


Figura 2.10 Implementación del bloque Qinv

Fuente: elaboración propia

Tras ello, tenemos el controlador proporcional para las velocidades de rotación u_4 y u_5, en el que se aplica la ley de control exactamente de la misma forma que para las posiciones angulares (Ecuación 2.19):

$$\dot{u_i^*} = K_u(u_i^* - u_i)$$ ***Ecuación 2.22***

La salida de dicho controlador (aceleración angular deseada o $\dot{u_i^*}$) debe ser convertida ahora a par aplicado T_1 y T_2. Esa es la tarea realizada en el bloque "D", el último dentro de "Rori" (o "Crot" en el desarrollo [2]). De acuerdo con lo explicado en el desarrollo de [1], el bloque "D" es un bloque de desacople entre los pares y las velocidades angulares, cuya implementación se realiza a través de técnicas conocidas en la teoría de control. En nuestro desarrollo, la elección de este desacople se ha realizado de tal forma que la traducción entre $u_{4,5}$ y $u_{4,5}^*$ se haga a través de integradores independientes. Esto es lo que permite la implementación del lazo interno con ganacia proporcional K_u que controla $u_{4,5}$. La estructura interna del bloque de desacople es la siguiente:

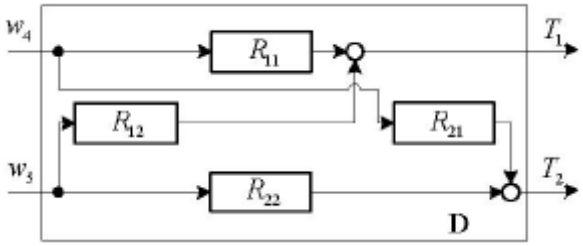


Figura 2.11 Estructura del bloque de desacoplo D

Fuente: [1]

Donde cada una de las R_{ij} se computa por medio de las ecuaciones de Euler para la dinámica rotacional (Ecuación 2.10 y Ecuación 2.11), resultando:

$$R_{11} = I_{11}$$ ***Ecuación 2.23***

$$R_{22} = I_{22}$$ ***Ecuación 2.24***

$$R_{12} = R_{21} = 0$$ ***Ecuación 2.25***

Para la obtención de dichos resultados se han despreciado los efectos de los términos $(I_{33} - I_{22})u_5u_6$ y $(I_{33} - I_{11})u_4u_6$, suponiendo que las funciones a implementar son $\dot{u_4^*} = \frac{T_1}{I_{11}}$ y $\dot{u_5^*} = \frac{T_2}{I_{22}}$. La implementación del bloque "D" en el modelo de Simulink, por lo tanto, se realizará mediante ganancias.

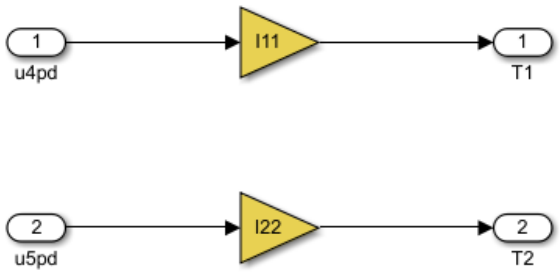


Figura 2.12 Implementación del bloque D

Fuente: elaboración propia

Hasta ahora hemos implementado todo lo que en la "Figura 2.3 Esquema de control realimentado" sería el controlador del sistema realimentado para la parte de rotación, *pitch* y *roll*. Como se ha explicado, ha sido a grandes rasgos un controlador basado en la inversión de las ecuaciones del modelo mecánico del vehículo autónomo.

2.2.2 Control de guiñada

En nuestro sistema, el *yaw* es la coordenada generalizada q_6, de la cual dijimos que su bucle de control debía ser mucho más rápido que el del resto, para poder realizar la importante aproximación de la Ecuación 2.9. La estructura del controlador, basándonos en el desarrollo del artículo [2], consta de dos controladores a su vez,

como se indica a continuación.

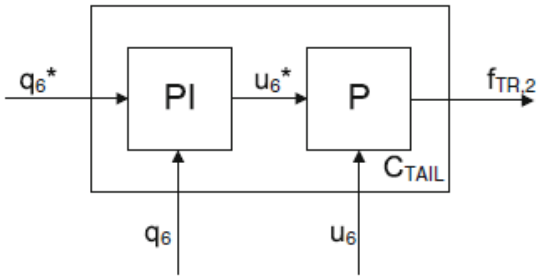


Figura 2.13 Esquema de control del yaw

Fuente: [2]

Un controlador PI que llevará la variable q_6 hacia su consigna mediante la actuación de u_6, y una ley proporcional simple que se encargue de que la velocidad de giro u_6 sea la adecuada para conseguir el control deseado en el primero. La variable $f_{TR,2}$ que se observa en la figura anterior no es otra cosa que el par aplicado en el eje vertical del sistema {f}, lo que en nuestra notación hemos llamado T_3 (Figura 2.1 Esquema de un UAV genérico).

2.2.3 Asignación de polos

En [2], tras una sintonización clásica de los parámetros de los controladores y tras la observación de las limitaciones físicas del sistema en los experimentos, se llegó a la conclusión de que todos los polos del sistema deberían ser colocados alrededor de -1, a excepción de los correspondientes a los de la dinámica del *yaw*. Estos últimos serían colocados en valores próximos a -10, y al ser su valor absoluto un orden de magnitud más grande que los otros, la dinámica de la orientación quedaría mucho más rápida que la de traslación, cumpliéndose la Ecuación 2.9.

Según el desarrollo de [2], las funciones de transferencia resultantes del sistema son las siguientes:

$$\frac{q_i}{q_i^*} = \frac{K_p^{q45}(K_I^{qi} + K_p^{qi}s)}{K_I^{qi}K_p^{q45} + K_p^{qi}K_p^{q45}s + K_D^{qi}K_p^{q45}s^2 + K_p^{q45}s^3 + K_p^{u45}s^4 + s^5} \quad i = 1,2$$

Ecuación 2.26

$$\frac{q_3}{q_3^*} = \frac{K_I^{q3} + K_p^{q3}s}{K_I^{q3} + K_p^{q3}s + K_D^{q3}s^2 + s^3}$$

Ecuación 2.27

$$\frac{q_6}{q_6^*} = \frac{K_p^{u6}\left(K_I^{q6} + K_p^{q6}s\right)}{K_I^{q6}K_p^{u6} + K_p^{u6}K_p^{q6}s + K_p^{u6}s^2 + s^3}$$

Ecuación 2.28

Cuya derivación permitió la obtención de las expresiones que se muestran a continuación para cada una de las constantes de los controladores.

$$K_p^{u45} = -(p1 + p2 + p3 + p4 + p5) \quad \textit{Ecuación 2.29}$$

$$\mathrm{K_p^{q45}} = p4p5 + p3(p4 + p5) + p2(p3 + p4 + p5) + p1(p2 + p3 + p4 + p5) \quad \textit{Ecuación 2.30}$$

$$Kp^{q45}K_D^{q12} = -p3p4p5 - p2(p3(p4 + p5) + p4p5) - p1(p2(p3 + p4 + p5) + p3(p4 + p5) + p4p5) \quad \textit{Ecuación 2.31}$$

$$K_p^{q45}K_p^{q12} = p2p3p4p5 + p1(p3p4p5 + p2(p3(p4 + p5) + p4p5)) \quad \textit{Ecuación 2.32}$$

$$K_p^{q45}K_I^{q12} = -p1p2p3p4p5 \quad \textit{Ecuación 2.33}$$

$$K_D^{q3} = -(p1 + p2 + p3) \quad \textit{Ecuación 2.34}$$

$$K_p^{q3} = p1(p2 + p3) + p2p3 \quad \textit{Ecuación 2.35}$$

$$K_I^{q3} = -p1p2p3 \quad \textit{Ecuación 2.36}$$

En nuestro caso los controladores del bucle del *yaw* tienen signo contrario que en [2].

$$K_p^{u6} = -(p6 + p7 + p8) \quad \textit{Ecuación 2.37}$$

$$K_p^{u6}K_p^{q6} = p6p7 + p8(p6 + p7)) \quad \textit{Ecuación 2.38}$$

$$K_p^{u6}K_I^{q6} = -p6p7p8 \quad \textit{Ecuación 2.39}$$

La situación de los polos es $p1 = p2 = p3 = p4 = p5 = -1$ en el caso de los polos para la traslación, y en los del segundo lazo será suficiente con situarlos tal que $p6 = p7 = p8 = -3$.

2.3 Simulación

Todo el modelo desarrollado en los epígrafes anteriores ha sido implementado en Simulink®, tal y como se muestra en el anexo A. A continuación, se realiza una simulación del sistema para verificar el funcionamiento del controlador.

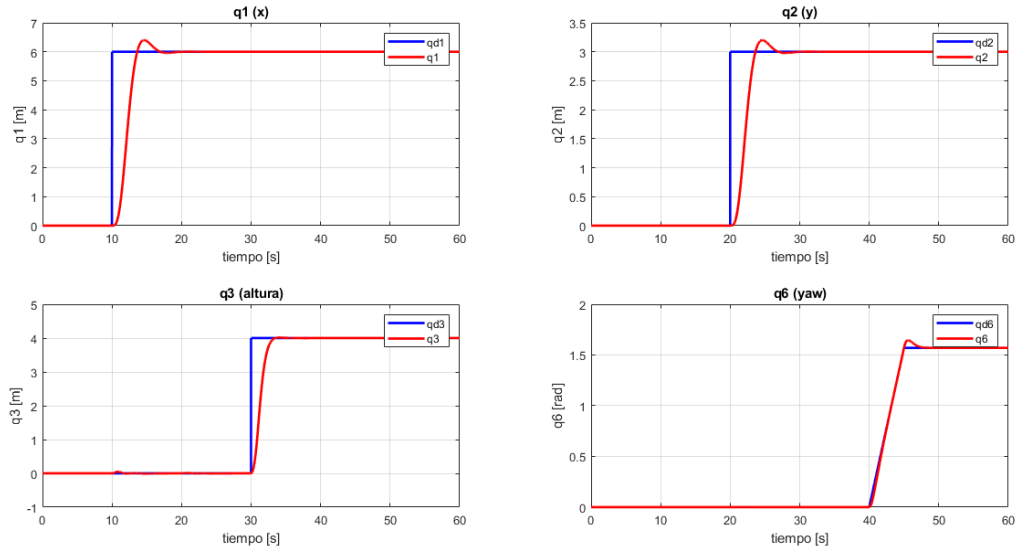


Figura 2.14 Simulación 3D – 1

Todas las señales siguen a su referencia, incluso teniendo en cuenta que las consignas en escalón son algo bruscas (sólo se ha considerado una entrada en forma de rampa para el *yaw*). Se puede observar que la situación de los polos elegida tiene un comportamiento ligeramente oscilatorio al alcanzar la referencia, comportamiento que debe tenerse en cuenta según la aplicación que quiera dársele al UAV. Se hace bastante notorio el efecto de la ación integral, ya que ésta es la responsable de la sobreoscilación antes mencionada, pero también de la ausencia de error en régimen permanente y de la gran exactitud con la que se sigue la referencia en rampa de q_6, como puede verse con un poco más de detalle a continuación.

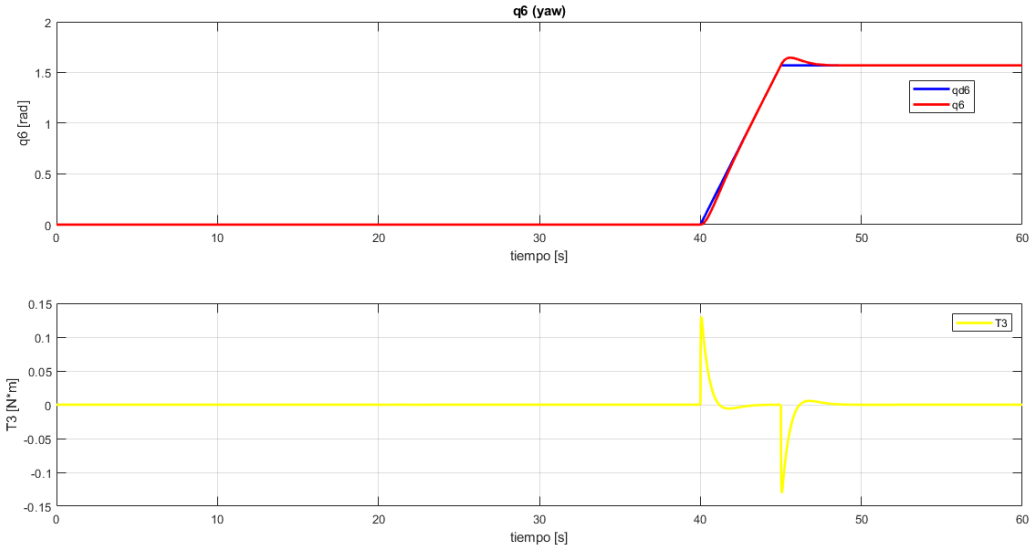


Figura 2.15 Simulación 3D – 2

En la anterior figura se muestra, además de la evolución del *yaw,* el par encargado de provocar ese giro, T_3.

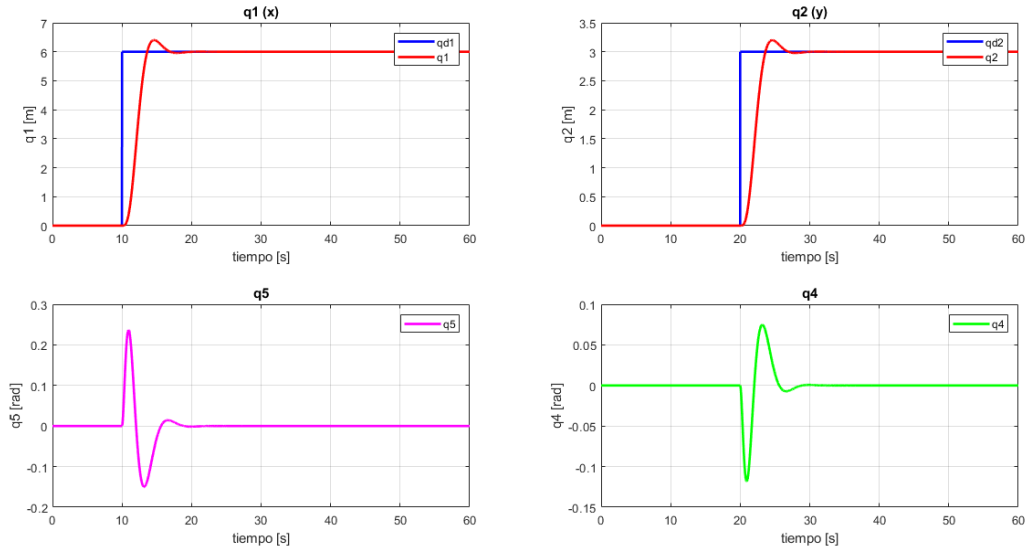


Figura 2.16 Simulación 3D – 3

Como se comentó anteriormente, los desplazamientos horizontales del quadrotor se consiguen inclinando la plataforma para crear una componente no nula de F_3 en el plano horizontal. En la figura anterior se puede observar dicha inclinación, de q_4 radianes en el eje "y" (o n_2) y q_5 en el eje "x" (n_1).

Por último, las cuatro variables de actuación:

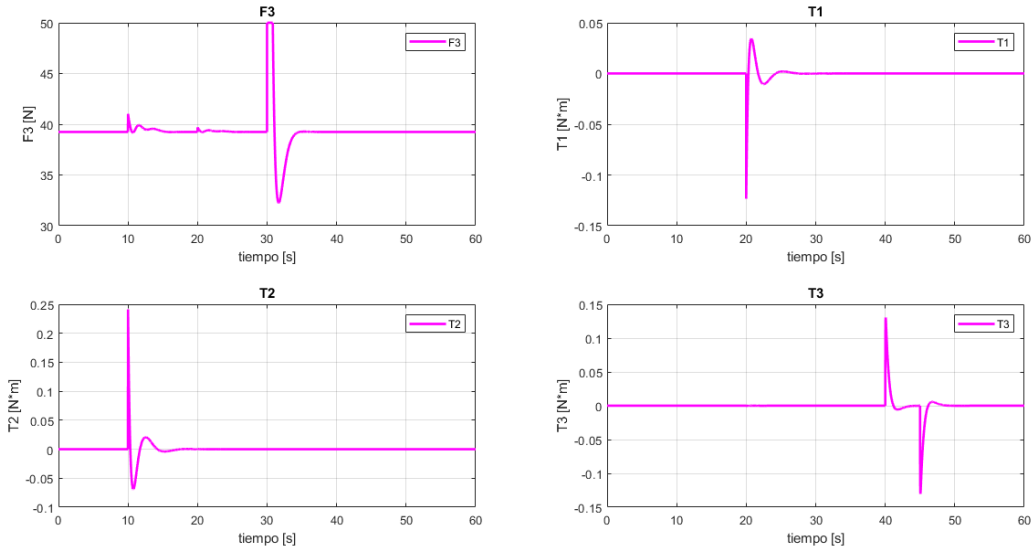


Figura 2.17 Simulación 3D – 4

Se puede observar una saturación de la fuerza vertical $F_{3,max} = 50\ N$; esto se debe a que en el modelo de Simulink se ha establecido como máximo este valor. En un sistema real, fuera de una simulación, la fuerza de sustentamiento máxima que debe poder alcanzar un quadrotor debe ser al menos del orden de dos veces su peso para poder volar y maniobrar correctamente. En este caso, al ser la masa de 4 Kg, la F_3 de saturación elegida debería haber sido aproximadamente $F_{3,max} \sim 2 * M * g \approx 80\ N$.

En términos generales, el comportamiento del sistema es muy exacto debido a que el controlador implementado contiene directamente la inversión del modelo simulado. Habría que disponer de un sistema un tanto más realista o diferente para observar realmente la robustez del controlador. Este aspecto es tratado en este trabajo mediante el modelado de sensores realistas y con ruido añadido.

3 MODELO DEL MANIPULADOR AÉREO

En esta sección se desarrolla el modelo físico tanto de la plataforma aérea como del brazo manipulador que le acoplaremos, y posteriormente el modelo conjunto de ambas. Para simplificar los cálculos, y puesto que las coordenadas q_4 y q_5 de la plataforma son el *pitch & roll,* que se controlan de la misma manera y pueden ser intercambiables (Figura 2.1 Esquema de un UAV genérico), eliminaremos una de estas dos. De esa manera, el movimiento del dron quedaría restringido al plano 2D vertical, al igual que el del manipulador, puesto que tiene únicamente 2 grados de libertad con vínculos rotativos.

Comenzaremos obteniendo el modelo de la plataforma teniendo en cuenta al brazo robótico; más tarde realizaremos el modelado del manipulador para poder elaborar su controlador y por último realizaremos simulaciones de ambos para comprobar su funcionamiento. Con el fin de poder calcular estos controladores y realizar las simulaciones, debemos definir cuantitativamente el sistema a modelar.

Parámetro	Valor	Uds	Descripción
g	9.81	m/s^2	Aceleración de la gravedad
M	4	Kg	Masa de la plataforma (UAV)
L1	0.3	m	Longitud eslabón 1 manipulador
L2	0.3	m	Longitud eslabón 2 manipulador
M1	0.15	Kg	Masa eslabón 1 manipulador
M2	0.15	Kg	Masa eslabón 2 manipulador
d	0	m	Distancia del centro de gravedad al primer vínculo del manipulador

Tabla 3-1 Parámetros y constantes del sistema

3.1 Modelo de la plataforma en 2D

En el apartado "2 Modelo y control tridimensional de la plataforma multirrotor", partimos desde las ecuaciones de la física ya resueltas por [1] para el sistema que pretendíamos modelar. En esta ocasión, realizaremos nuestro propio desarrollo para la obtención de las ecuaciones mecánicas del sistema. El esquema de la plataforma en 2D que queremos modelar es el siguiente, donde los sistemas de referencia y la nomenclatura de las variables es exactamente igual que en el epígrafe antes mencionado.

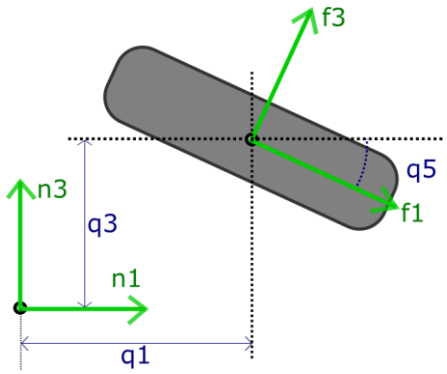


Figura 3.1 Plataforma 2D

Fuente: elaboración propia

{n} es el sistema de referencia fijo a la tierra, mientras que son los ejes {f} los solidarios al dron. Las posiciones generalizadas q_1 y q_3 nombran, respectivamente, la posición del centro de masas del UAV con respecto al origen de n_1 y n_3, posición longitudinal y altura. El giro del cuerpo móvil con respecto a {n} viene dado por el ángulo q_5, con sentido positivo el indicado en la figura. Actúa la aceleración de la gravedad en el sentido negativo del eje n_3, y las otras variables de actuación dinámicas del sistema serán el par T_2, aplicado en el eje f_2 y responsable de la orientación de q_5, y la fuerza de sustentamiento F_3, aplicada en el eje f_3.

3.1.1 Obtención mediante ecuaciones de la mecánica

Al ser {n} un sistema de referencia inercial, las ecuaciones para la cinemática, tanto de traslación como de rotación, no van a variar con respecto a las presentadas en el modelo 3D del vehículo. La velocidad lineal de un cuerpo es la derivada con respecto al tiempo de su posición espacial:

$$\boldsymbol{v} = \frac{d\boldsymbol{x}}{dt}$$

Por tanto, presentando la ecuación anterior en función de las variables generalizadas de nuestro sistema, obtenemos:

$$\dot{q_1} = u_1 \qquad \textit{Ecuación 3.1}$$

$$\dot{q_3} = u_3 \qquad \textit{Ecuación 3.2}$$

La velocidad angular se define como la derivada temporal de la posición angular con respecto al tiempo:

$$\boldsymbol{\omega} = \frac{d\boldsymbol{\theta}}{dt}$$

Lo que nos proporciona la tercera ecuación, y última de la cinemática directa del sistema.

$$\dot{q_5} = u_5 \qquad \textit{Ecuación 3.3}$$

En cuanto a la dinámica, podemos aplicar los teoremas de la mecánica del sólido rígido, el Teorema de la Cantidad de Movimiento (TCM), y el Teorema del Momento Cinético (TMC), que unidos a las ecuaciones horarias de la cinemática, son suficientes para identificar completamente el sistema.

El primero, el TCM, no es más que otra forma de llamar a la segunda ley de Newton:

$$\frac{d\boldsymbol{p}}{dt} = M\frac{d\boldsymbol{v}}{dt} = M\boldsymbol{a} = \sum \boldsymbol{F}_{ext}$$

Donde **p** es el vector de momento lineal, **v** la velocidad, M la masa del sistema, **a** la aceleración y el último término es el sumatorio de todas las fuerzas externas al sistema.

La particularización de esta ecuación vectorial en nuestro sistema nos proporciona dos ecuaciones, a lo largo de

los ejes n_1 y n_2.

$$\ddot{u}_1 M = F_3 \sin(q_5) \qquad \textit{Ecuación 3.4}$$

$$\ddot{u}_3 M = F_3 \cos(q_5) - Mg \qquad \textit{Ecuación 3.5}$$

Solo nos falta aplicar el TMC, cuya ecuación es:

$$\frac{d\boldsymbol{L}_G}{dt}\Big)_N = \sum \boldsymbol{M}_{ext}$$

Donde $\sum \mathbf{M}_{ext}$ es el sumatorio de torques externos aplicados al sistema, y $\mathbf{L}_g$ es el momento angular con respecto al centro de masas, cuya expresión también se conoce:

$$\boldsymbol{L}_G = \boldsymbol{I}_G \boldsymbol{\omega}_{F-N} = \begin{bmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

La expresión del momento cinético particularizada en nuestro sistema sólo tiene compenente en el eje f_2:

$$\boldsymbol{L}_G = I_{22} u_5 \boldsymbol{f_2}$$

A continuación, hemos de notar que la expresión del momento cinético que hemos calculado está expresada en los ejes {f}, mientras que en el TMC está derivado en el marco de referencia {n}. Para solucionar este problema, aplicamos la Regla de Poisson:

$$\frac{d\boldsymbol{L}_G}{dt}\Big)_N = \frac{d\boldsymbol{L}_G}{dt}\Big)_F + \boldsymbol{\omega_{F-N}} \times \boldsymbol{L}_G$$

Si calculamos el producto vectorial en nuestro sistema por medio del determinante, comprobamos que resulta nulo.

$$\boldsymbol{\omega_{F-N}} \times \boldsymbol{L}_G = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ 0 & u_5 & 0 \\ 0 & I_{22} u_5 & 0 \end{vmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Por lo que la derivada del momento cinético queda:

$$\frac{d\boldsymbol{L}_G}{dt}\Big)_N = \frac{d\boldsymbol{L}_G}{dt}\Big)_F = I_{22} \dot{u}_5 \boldsymbol{f_2}$$

Finalmente, aplicando el TMC tenemos la última ecuación:

$$I_{22} \dot{u}_5 = T_2 \qquad \textit{Ecuación 3.6}$$

3.1.2 Obtención mediante particularización de las ecuaciones 3D

La elección de los nombres de cada variable en el esquema de la plataforma (Figura 3.1 Plataforma 2D) no es aleatoria. Están puestas exactamente igual que en esquema 3D tomado de [1] (Figura 2.1 Esquema de un UAV genérico), puesto que, si nuestros cálculos son correctos, las ecuaciones obtenidas en el epígrafe anterior deberían coincidir con las presentadas en el artículo antes mencionado. Para ello, la particularización que debemos hacer en las ecuaciones 3D del epígrafe "2.1 Modelo" de esta memoria, es la siguiente:

$$q_2 = q_4 = q_6 = u_2 = u_4 = u_6 = T_1 = T_3 = F_1 = F_2 = 0 \qquad \textit{Ecuación 3.7}$$

Que no es otra cosa que restringir el movimiento del sistema al plano vertical 2D. Se comprueba que haciendo tal particularización de las ecuaciones presentadas en "2.1 Modelo", se obtienen las siguientes.

$$\dot{q_1} = u_1$$ ***Ecuación 3.8***

~~$\dot{q_2} = u_2$~~ ***Ecuación 3.9***

$$\dot{q_3} = u_3$$ ***Ecuación 3.10***

~~$\dot{q_4} = (u_4 \cos(q_6) - u_5 \sin(q_6))/\cos(q_5)$~~ ***Ecuación 3.11***

$$\dot{q_5} = \text{~~}u_4 \sin(q_6)\text{~~} + u_5 \cos(\text{~~}q_6\text{~~}) \rightarrow \dot{q_5} = u5$$ ***Ecuación 3.12***

~~$\dot{q_6} = u_6 + \tan(q_5)(u_4 \cos(q_6) - u_5 \sin(q_6))$~~ ***Ecuación 3.13***

$$\begin{pmatrix} \dot{u_1} \\ \text{~~}\dot{u_2}\text{~~} \\ \dot{u_3} \end{pmatrix} M = \boldsymbol{C_{F-N}} \begin{pmatrix} \text{~~}F_1\text{~~} \\ \text{~~}F_2\text{~~} \\ F_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -g * M \end{pmatrix}$$ ***Ecuación 3.14***

~~$T1 + (I_{33} - I_{22})u_5 u_6 - I_{11}\dot{u_4} = 0$~~ ***Ecuación 3.15***

$$T2\text{~~} - (I_{33} - I_{11})u_4 u_6\text{~~} - I_{22}\dot{u_5} = 0 \rightarrow T_2 = I_{22}\dot{u_5}$$ ***Ecuación 3.16***

~~$T3 + (I_{22} - I_{11})u_4 u_5 - I_{33}\dot{u_6} = 0$~~ ***Ecuación 3.17***

Recalculamos la matriz de rotación.

$$\boldsymbol{C_{F-N}} = \begin{bmatrix} cos(q_5) & 0 & sin(q_5) \\ 0 & 1 & 0 \\ -sin(q_5) & 0 & cos(q_5) \end{bmatrix}$$

Con lo que la ecuación vectorial "Ecuación 3.14" se divide en dos ecuaciones escalares.

$$\dot{u_1} M = F_3 \sin(q_5)$$ ***Ecuación 3.18***

$$\dot{u_3} M = F_3 \cos(q_5) - Mg$$ ***Ecuación 3.19***

Concluimos que el resultado de la particularización de las ecuaciones del artículo [1] para el caso 2D es exactamente igual a lo obtenido en nuestro desarrollo de las ecuaciones de la mecánica.

3.2 Modelo del brazo manipulador

Como ya se ha dicho anteriormente, el brazo robótico que vamos a acoplar en nuestro robot va a ser un manipulador formado por dos eslabones rígidos con forma de barra, unidos entre sí y a la plataforma mediante dos vínculos rotativos simples. Esta configuración, con dos grados de libertad, permite únicamente el movimiento del brazo en el plano 2D, el plano vertical en el que se mueve el vehículo. Si se deseara estudiar el caso tridimensional, sería cuestión de añadir, por ejemplo, una rotación extra en la base del brazo que permitiera girar su plano de movimiento libremente. Así, el extremo del manipulador, donde se situaría la herramienta podría alcanzar puntos en todas las direcciones sin la necesidad de rotar la plataforma completa. No es lo mismo que si quisiéramos tener la posibilidad de alcanzar cualquier punto con el efector final orientado en cualquier dirección; para ello, en el caso 3D, necesitaríamos un manipulador completo, con articulaciones que permitieran tantos giros como grados de libertad (6 en total).

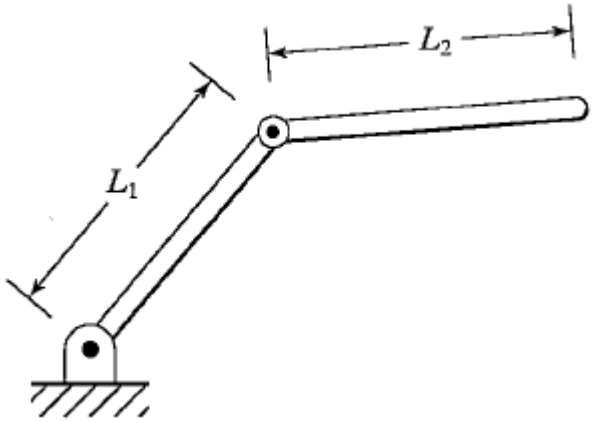


Figura 3.2 Esquema del brazo manipulador

Fuente: [8]

El esquema de la figura anterior muestra un mecanismo con la misma estructura que nuestro manipulador. Dos eslabones unidos entre sí y al suelo por medio de rotaciones simples. En nuestro caso, la diferencia principal es que el primer eslabón no estará vinculado al suelo, sino a la parte inferior del vehículo que se describía en el apartado "3.1 Modelo de la plataforma en 2D". Si colocáramos el brazo en esa posición, tendríamos la estructura mostrada en la "Figura 1.7 UAV + manipulador".

3.2.1 Cinemática directa: Denavit-Hartemberg

A continuación, utilizaremos el algoritmo de Denavit-Hartemberg (D-H) en su modalidad estándar para obtener las matrices de trasnformación homogénea del brazo manipulador aislado. Con ello, habremos obtenido las ecuaciones que rigen la cinemática directa del mecanismo para su posterior utilización en el control del sistema completo. Es bien sabido que en un entorno n-dimensional las matrices de transformación son cuadradas y te tamaño 4x4.

$$T_{ab} = \begin{bmatrix} C11 & C12 & C13 & p1 \\ C21 & C22 & C23 & p2 \\ C31 & C32 & C33 & p3 \\ f1 & f2 & f3 & e \end{bmatrix}$$ ***Ecuación 3.20***

Donde C_{3x3} es la matriz de orientación del punto b respecto al punto a, p_{3x1} es el vector de desplazamiento de b respecto a, f_{1x3} es un vector de perspectiva y e_{1x1} es un factor de escala. Las matrices de transformación homogénea son una herramienta matemática muy útil en este tipo de estudios, y los pasos a seguir para encontrar la matriz de transformación entre la base y el efector final son los siguientes:

1. Encontrar los parámetros D-H de cada una de las articulaciones.
2. Calcular, a partir de los parámetros, las matrices T_{i-i+1}
3. Calcular la matriz $T_{0-n} = \prod T_{i-i+1}$

Por lo que comenzamos con el primer paso: el algoritmo de D-H. [7]

- D-H 1.- Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.

En nuestro caso no existe base fija, sino que el eslabón 0 corresponde a la parte inferior de la plataforma en la que se acopla el manipulador.

- D-H 2.- Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n.

El primer vínculo rotativo es la articulación 1, y el segundo la articulación 2.

- D-H 3.- Localizar el eje de cada articulación. Si esta es rotativa, el eje será su propio eje de giro. Si es

prismática, será el eje a lo largo del cual se produce el desplazamiento.

- D-H 4.- Para i de 0 a n-1 situar el eje zi sobre el eje de la articulación i+1.

Como sólo tenemos dos articulaciones de rotación, situaremos el eje Z_0 sobre el eje de rotación 1 y Z_1 sobre el eje de rotación de la articulación 2.

- D-H 5.- Situar el origen del sistema de la base {S0} en cualquier punto del eje z_0. Los ejes x_0 e y_0 se situarán de modo que formen un sistema dextrógiro con z_0.
- D-H 6.- Para i de 1 a n-1, situar el sistema $\{S_i\}$ (solidario al eslabón i) en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i. Si ambos ejes se cortasen se situaría $\{S_i\}$ en el punto de corte. Si fuesen paralelos $\{S_i\}$ se situaría en la articulación i+1.
- D-H 7.- Situar x_i en la línea normal común a z_{i-1} y z_i
- D-H 8.- Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i.
- D-H 9.- Situar el sistema $\{S_n\}$ en el extremo del robot de modo que zn coincida con la dirección de z_{n-1} y x_n sea normal a z_{n-1} y z_n.

El resultado de los pasos 1 a 9 se puede comprobar en la siguiente figura.

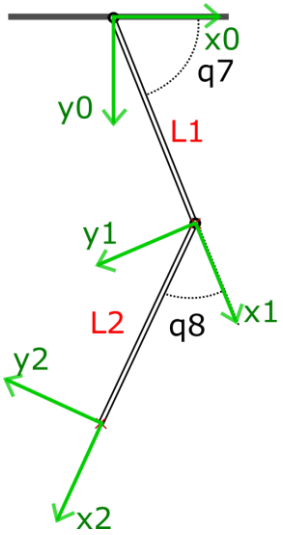


Figura 3.3 Manipulador con ejes de D-H

Fuente: elaboración propia

A continuación, obtendremos los parámetros para poder calcular la matriz de transformación.

- D-H 10.- Obtener θ_i como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos.
- D-H 11.- Obtener d_i como la distancia, medida a lo largo de z_{i-1}, que habría que desplazar $\{S_{i-1}\}$ para que x_i y x_{i-1} quedasen alineados.
- DH 12.- Obtener a_i como la distancia medida a lo largo de x_i (que ahora coincidiría con x_{i-1}) que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con $\{S_i\}$.
- DH 13.- Obtener α_i como el ángulo que habría que girar entorno a x_i (que ahora coincidiría con x_{i-1}), para que el nuevo $\{S_{i-1}\}$ coincidiese totalmente con $\{S_i\}$.

El resultado se resume en la siguiente tabla:

i	ϴ	d	a	α
1	**q_1**	**0**	**L_1**	**0**
2	**q_2**	**0**	**L_2**	**0**

Tabla 3-2 Parámetros de D-H

Una vez que tenemos los parámetros de D-H, buscamos cada una de las matrices de transformación $T_{i,i+1}$ o matrices de D-H (MDH), cuya expresión es conocida:

$$MDH = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textit{Ecuación 3.21}$$

En el Anexo B se muestra el código con el que se calculan cada una de las matrices $T_{0,1}$ y $T_{1,2}$, así como la matriz de transformación completa $T_{0,ee}$, donde ee denota al efector final. El resultado de dichos cálculos es el siguiente:

$$T_{0-2} = \begin{bmatrix} cos(q7+q8) & -sin(q7+q8) & 0 & l2*cos(q7+q8)+l1*cos(q7) \\ sin(q7+q8) & cos(q7+q8) & 0 & l2*sin(q7+q8)+l1*sin(q7) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textit{Ecuación 3.22}$$

Por lo que las ecuaciones de la cinemática directa son:

$$\begin{cases} x_{ee} = L_2\cos(q_7+q_8) + L_1\cos(q_7) \\ y_{ee} = L_2\sin(q_7+q_8) + L_1\sin(q_7) \end{cases} \quad \textit{Ecuación 3.23}$$

3.2.2 Cinemática Inversa

Con las ecuaciones del apartado anterior podemos conocer la posición del efector final del brazo a partir del conocimiento de la posición de cada una de las articulaciones (q_7 y q_8). En este trabajo desarrollaremos un controlador independiente para cada articulación, y en vista al posible control cinemático con referencias en coordenadas cartesianas en lugar de articulares, necesitamos plantear el problema cinemático inverso.

El problema de la cinemática inversa, al ser en general el sistema estudiado altamente no lineal, suele no estar bien definido o presentar múltiples soluciones, condiciones que dificultan el proceso de obtención de posiciones articulares.

Partiendo de las ecuaciones de la cinemática directa obtenidas en el epígrafe anterior (Ecuación 3.23), elevamos cada una de ellas al cuadrado y las sumamos.

$$x^2 + y^2 = L_2^2 + L_1^2 + 2L_1L_2[C_7 * C_{7+8} + S_7S_{7+8}]$$

Dentro del corchete podemos aplicar la fórmula trigonométrica del coseno de la diferencia ($C_{a-b} = C_aC_b + S_aS_b$), con lo que queda:

$$x^2 + y^2 = L_2^2 + L_1^2 + 2L_1L_2C_8$$

De donde podemos obtener el valor de q_8:

$$q_8 = atan2\left(\pm\sqrt{1-\left(\frac{x^2+y^2-L_1^2-L_2^2}{2L_1L_2}\right)^2}, \frac{x^2+y^2-L_1^2-L_2^2}{2L_1L_2}\right) \quad \textit{Ecuación 3.24}$$

Usamos el atan2 de Matlab en lugar de un arcotangente normal para evitar indeterminaciones de cuadrante.

También debemos notar que existen dos soluciones en la expresión anterior, de las cuales nos quedamos con aquella que resulta al elegir el signo positivo. Estas soluciones corresponden a la configuración de "codo hacia arriba" o "codo hacia abajo", y en una aplicación real se seleccionaría atendiendo a criterios de necesidad, eligiendo aquella que evitara colisiones con objetos del entorno.

Para q_7, ahora que se conoce el valor de q_8 tomamos la primera ecuación de la cinemática directa (la de x) y volvemos a aplicar una formula trigonométrica, pero esta vez el coseno de la suma ($C_{7+8} = C_7C_8 - S_7S_8$), y nos queda lo siguiente:

$$x = L_2(C_7C_8 - S_7S_8) + L_1C_7$$

Expresión en la cual sacamos factor común cos(q_7) y aplicamos el siguiente cambio de variables:

$$\begin{Bmatrix} \rho\cos(\alpha) = L_2C_8 + L_1 \\ \rho\sin(\alpha) = L_2S_8 \end{Bmatrix}$$

Entonces la ecuación de x queda:

$$x = C_7\rho C_\alpha - S_7\rho S_\alpha$$

Si dividimos la ecuación entre ρ y aplicamos la fórmula del coseno de la suma para q_7 y α, podemos calcular finalmente q_7:

$$q_7 + \alpha = atan2(\pm\sqrt{1 - \left(\frac{x}{\rho}\right)^2}, \frac{x}{\rho}) \qquad \textit{Ecuación 3.25}$$

Y nos quedamos de nuevo con la solución correspondiente al signo positivo.

3.2.3 Dinámica

Tanto para el problema de la obtención del modelo dinámico (ecuaciones de movimiento) como el de control del manipulador, vamos a seguir el desarrollo de los capítulos 6 y 10 de [8]. En ellos se detalla el proceso de modelado y control de un robot manipulador industrial genérico por el método del Par Computado.

En el capítulo 6 antes mencionado, se trata de obtener las ecuaciones en forma cerrada para la dinámica de un manipulador o "ecuaciones en el espacio de estados", llamadas así por la dependencia de sus términos tanto de q como de $\dot{q}$. Para un brazo sobre el que actuaremos mediante pares aplicados en las articulaciones, pueden escribirse de la siguiente forma:

$$\tau = M(q)\ddot{q} + V(q,\dot{q}) + G(q) + F(q,\dot{q}) \qquad \textit{Ecuación 3.26}$$

Donde tau es la señal de actuación (par aplicado en los motores de las articulaciones), M es la matriz de inercias, V es un vector de términos centrífugos y de Coriolis, G es un vector con términos gravitatorios y F es un vector que modela la fricción y otros efectos. Utilizaremos el conocido método recursivo de Newton-Euler (N-E) para obtener las expresiones de cada uno de los pares aplicados en los motores, T_7 y T_8, cuyo procedimiento también aparece detallado en [8].

Antes de continuar con las ecuaciones de N-E, no podemos olvidar mencionar que, en nuestro manipulador, vamos a modelar las masas de los eslabones como masas puntuales concentradas en el extremo de cada eslabón; es decir, que la masa del eslabón 1 está concentrada en la articulación que une a los eslabones 1 y 2, mientras que la masa del eslabón 2 se encuentra en el punto correspondiente al efector final del brazo.

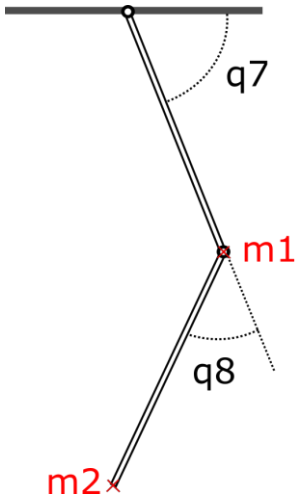


Figura 3.4 Situación de las masas de los eslabones

Fuente: elaboración propia

El sistema de ejes elegido para este algoritmo (N-E) es el que se representa en la siguiente figura.

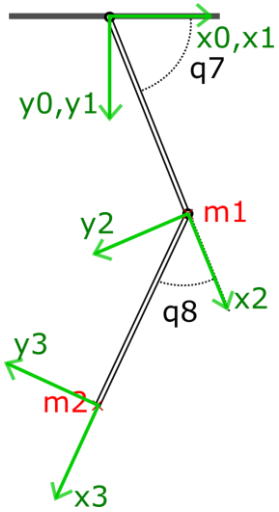


Figura 3.5 Manipulador con ejes de N-E

Fuente: elaboración propia

El centro de gravedad de cada uno de los eslabones se encuentra sobre el punto de masa correspondiente, y por el hecho de tener la masa concentrada nos encontramos que los tensores de inercia escritos sobre el centro de gravedad son nulos.

$$I_1(CG_1) = I_2(CG_2) = 0$$

No existe fuerza ni par aplicado sobre el efector final.

$$f_{ee} = n_{ee} = 0$$

Para el diseño suponemos que no existe rotación de la base, simplificación que será correcta siempre que se realicen movimientos suaves de forma que el ángulo q_5 sea pequeño, como las velocidades y aceleraciones traslacionales.

$$\omega_0 = \dot{\omega_0} = 0$$

Se incluye la aceleración de la gravedad.

$$\dot{v_{0,0}} = g\boldsymbol{y_o}$$

Se calculan las matrices de rotación entre los sistemas de referencia sucesivos.

$$R_{i,i+1} = \begin{bmatrix} c_{i+1} & -s_{i+1} & 0 \\ s_{i+1} & c_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{i+1,i} = \begin{bmatrix} c_{i+1} & s_{i+1} & 0 \\ -s_{i+1} & c_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Iteración hacia fuera para el eslabón 1:

$$\omega_{1,1} = \dot{q_7}\mathbf{z_1}$$

$$\dot{\omega_{1,1}} = \ddot{q_7}\mathbf{z_1}$$

$$\dot{v_{1,1}} = gS_7\mathbf{x_1} + gC_7\mathbf{y_1}$$

$$\dot{v_{1,cg1}} = \begin{bmatrix} 0 \\ l1\ddot{q_7} \\ 0 \end{bmatrix} + \begin{bmatrix} -l1\dot{q_7}^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} gS_7 \\ gC_7 \\ 0 \end{bmatrix} = \begin{bmatrix} -l1\dot{q_7}^2 + gS_7 \\ l_1\ddot{q_7} + gC_7 \\ 0 \end{bmatrix}$$

$$F_{1,1} = \begin{bmatrix} -m_1 l_1 \dot{q_7}^2 + m_1 g S_7 \\ m_1 l_1 \ddot{q_7} + m_1 g C_7 \\ 0 \end{bmatrix}$$

$$N_{1,1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

- Iteración hacia fuera para el eslabón 2:

$$\omega_{2,2} = \begin{bmatrix} 0 \\ 0 \\ \dot{q_7} + \dot{q_8} \end{bmatrix}$$

$$\dot{\omega}_{2,2} = \begin{bmatrix} 0 \\ 0 \\ \ddot{q_7} + \ddot{q_8} \end{bmatrix}$$

$$v_{2,2} = \begin{bmatrix} l_1\ddot{q_7}S_8 - l_1\dot{q_7}^2C_8 + gS_{78} \\ l1\ddot{q_7}C_8 + l_1\dot{q_7}^2S_8 + gC_{78} \\ 0 \end{bmatrix}$$

$$v_{2,c2} = \begin{bmatrix} 0 \\ l_2(\ddot{q_7}+\ddot{q_8}) \\ 0 \end{bmatrix} + \begin{bmatrix} -l_2(\dot{q_7} + \dot{q_8})^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} l_1\ddot{q_7}S_8 - l_1\dot{q_7}^2C_8 + gS_{78} \\ l_1\ddot{q_7}C_8 + l_1\dot{q_7}^2S_8 + gC_{78} \\ 0 \end{bmatrix}$$

$$F_{2,2} = \begin{bmatrix} m_2 l_1 \ddot{q_7} S_8 - m_2 l_1 \dot{q_7}^2 C_8 + m_2 g S_{78} - m_2 l_2 (\dot{q_7} + \dot{q_8})^2 \\ m_2 l_1 \ddot{q_7} C_8 + m_2 l_1 \dot{q_7}^2 S_8 + m_2 g C_{78} + m_2 l_2 (\ddot{q_7} + \ddot{q_8}) \\ 0 \end{bmatrix}$$

$$N_{2,2} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

- Iteración hacia dentro para el eslabón 2:

$$f_{2,2} = F_{2,2}$$

$$n_{2,2} = \begin{bmatrix} 0 \\ 0 \\ m_2 l_1 l_2 C_2 \ddot{q_7} + m_2 l_1 l_2 S_2 \dot{q_7}^2 + m_2 l_2 g C_{78} + m_2 l_2^2 (\ddot{q_7} + \ddot{q_8}) \end{bmatrix}$$

- Iteración hacia dentro para el eslabón 1:

$$f_{1,1} = \begin{bmatrix} C_8 & -S_8 & 0 \\ S_8 & C_8 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} m_2 l_1 \ddot{q_7} S_8 - m_2 l_1 \dot{q_7}^2 C_8 + m_2 g S_{78} - m_2 l_2 (\dot{q_7} + \dot{q_8})^2 \\ m_2 l_1 \ddot{q_7} C_8 + m_2 l_1 \dot{q_7}^2 S_8 + m_2 g C_{78} + m_2 l_2 (\ddot{q_7} + \ddot{q_8}) \\ 0 \end{bmatrix} + \begin{bmatrix} -m_1 l_1 \dot{q_7}^2 + m_1 g S_7 \\ m_1 l_1 \ddot{q_7} + m_1 g C_7 \\ 0 \end{bmatrix}$$

$$n_{1,1} = \begin{bmatrix} 0 \\ 0 \\ m_2 l_1 l_2 C_8 \ddot{q_7} + m_2 l_1 l_2 S_8 \dot{q_7}^2 + m_2 l_2 g C_{78} + m_2 l_2^2 (\ddot{q_7} + \ddot{q_8}) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m_1 l_1^2 \ddot{q_7} + m_1 l_1 g C_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m_2 l_1^2 \ddot{q_7} - m_2 l_1 l_2 S_8 (\dot{q_7} + \dot{q_8})^2 + m_2 l_1 g S_8 S_{78} + m_2 l_1 l_2 C_8 (\ddot{q_7} + \ddot{q_8}) + m_2 l_1 g C_8 C_{78} \end{bmatrix}$$

Extrayendo las componentes **z** de cada $n_{i,i}$, tenemos la ecuación de cada uno de los pares.

$$\tau_7 = m_2 l_2^2 (\ddot{q_7} + \ddot{q_8}) + m_2 l_1 l_2 C_8 (2\ddot{q_7} + \ddot{q_8}) + (m_1 + m_2) l_1^2 \ddot{q_7} - m_2 l_1 l_2 S_8 \dot{q_8}^2 - 2 m_2 l_1 l_2 S_8 \dot{q_7} \dot{q_8} + m_2 l_2 g C_{78} + m_2 l_2^2 (\ddot{q_7} + \ddot{q_8}) \quad \textit{Ecuación 3.27}$$

$$\tau_8 = m_2 l_1 l_2 C_2 \ddot{q_7} + m_2 l_1 l_2 S_2 \dot{q_7}^2 + m_2 l_2 g C_{78} + m_2 l_2^2 (\ddot{q_7} + \ddot{q_8}) \quad \textit{Ecuación 3.28}$$

Para sernos de utilidad, debemos reescribir las expresiones de ambos pares en función de las matrices M, V y G, tal y como se muestra en la Ecuación 3.26. Para ello, identificamos los términos comparando la expresión de τ con la ecuación en forma cerrada de la dinámica.

- Todo aquello que multiplique a $\ddot{q}_i$ forma parte de la matriz M.

$$M(q) = \begin{bmatrix} l_2^2 m_2 + 2 l_1 l_2 m_2 C_8 + l_1^2 (m_1 + m_2) & l_2^2 m_2 + l_1 l_2 m_2 C_8 \\ l_2^2 m_2 + l_1 l_2 m_2 C_8 & l_2^2 m_2 \end{bmatrix} \quad \textit{Ecuación 3.29}$$

Cabe mencionar que cualquier matriz M en este tipo de modelado de manipuladores, debe ser simétrica y definida positiva, y por lo tanto también invertible.

- Todos los términos restantes que dependan de $\dot{q}_i$ pertenecen al vector V.

$$V(q, \dot{q}) = \begin{bmatrix} -m_2 l_1 l_2 S_8 \dot{q_8}^2 - 2 m_2 l_1 l_2 S_8 \dot{q_7} \dot{q_8} \\ m_2 l_1 l_2 S_8 \dot{q_7}^2 \end{bmatrix} \quad \textit{Ecuación 3.30}$$

Dentro del vector V se agrupan términos resultantes de los esfuerzos centrífugos (todos aquellos que lleven la velocidad al cuadrado) como aquellos provenientes de las aceleraciones de Coriolis (aquellos que contengan productos de dos velocidades).

- Por último, en G agrupamos todos aquellos términos que multipliquen a g:

$$G(q) = \begin{bmatrix} m_2 l_2 g C_{78} + (m_1 + m_2) l_1 g C_7 \\ m_2 l_2 g C_{78} \end{bmatrix} \quad \textit{Ecuación 3.31}$$

3.3 Modelo del manipulador acoplado en la plataforma

En este epígrafe vamos a obtener el modelo completo del manipulador aéreo, es decir, la plataforma o UAV con el brazo manipulador acoplado en la parte inferior de su base. Cuando los problemas físicos adquieren complejidad y superan cierto número de grados de libertad, es frecuente recurrir a software, que de forma más secilla permiten aplicar las ecuaciones de la física clásica en la obtención de modelos. Autolev® es un software de modelado diámico simbólico, lo que nos permite obtener las expresones analíticas de las ecuaciones de movimiento (dinámica) de un sistema físico. Funciona a través de una interfaz de consola de comandos, en la cual se introduce un fichero de texto que contiene la definición del sistema a modelar en un lenguaje de programación propio. Tras unos instantes, el programa genera en el directorio otro fichero de texto.all de salida que contiene las ecuaciones de la dinámica en función de los parámetros establecidos, que ha calculado según

el método que elijamos (en nuestro caso las ecuaciones de Kane). También se puede escribir un comando para que el propio programa genere un fichero .m con las ecuaciones implementadas en la forma que las necesitamos para simular el sistema en Simulink (ecuaciones de movimiento $\ddot{q}_i = f(q_i, \dot{q}_i, actuación, parámetros)$.

En el anexo C puede verse el texto que contiene la declaración del sistema completo y una serie de comentarios añadidos que explican cada comando contenido. También se muestra todo el sistema que se ha declarado en el programa en la siguiente figura.

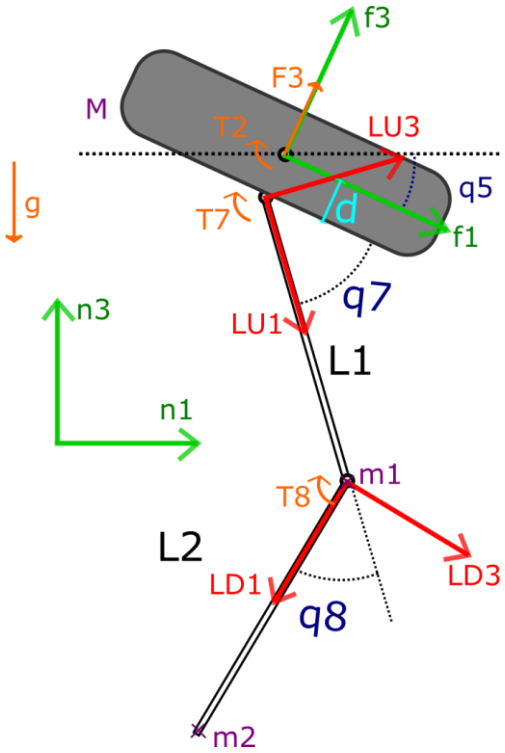


Figura 3.6 Sistema completo modelado en Autolev

Fuente: elaboración propia

Nota: Se ha realizado un cambio de ejes con respecto a los utilizados en el modelado del manipulador (ver Figura 3.5 Manipulador con ejes de N-E). Ahora las coordenadas que antes llamábamos "x" se han convertido en "1", y las "y" se han convertido en "3". Además, los ejes "3" están definidos en sentido hacia arriba (hacen las veces de Z) según {n} sin tener en cuenta los giros, mientras que en modelo de N-E los elegimos hacia abajo al declarar la gravedad positiva. Las matrices del modelo de ecuaciones de espacio-estado M y V se mantienen igual, salvo la consideración de que lo que antes eran componentes "y" ahora son componentes de los ejes "3". Los valores absolutos de las componentes del vector G son también los mismos, cambiando únicamente el signo al haber elegido el sentido contrario en el eje "3".

El valor cero para la coordenada de giro q_7 se encuentra con los ejes {LU} paralelos a los {f}, mientas que el origen de q_8 se encuentra con el segundo eslabón alineado con el primero. Las cuatro variables con las que el controlador actuará sobre el sistema son la fuerza de sustentamiento F_3 y los pares T_2 para girar la plataforma y T_7 y T_8 para los motores de las articulaciones.

Autolev genera también como salida el texto contenido en la segunda parte del anexo C, en el cual se encuentran las ecuaciones de movimiento en un formato muy parecido al que utilizaremos más tarde en la implementación en Simulink. Entre las líneas de dicho archivo se encuentra el problema dinámico resuelto mediante un sistema de ecuaciones lineales. Primero se calcula una serie de coeficientes que no son constantes, pero sí conocidos en cada paso de simulación. A continuación, se resuelve un sistema matricial de la forma $Ax = B$, que da solución a cada una de las variables de movimiento (aceleraciones de todas las variables generalizadas q_i).

4 CONTROL Y SIMULACIÓN DEL MANIPULADOR AÉREO SIN SENSORIZACIÓN

La forma de actuar para obtener un controlador para el sistema completo será la siguiente: primero desarrollaremos el controlador para la plataforma en 2D a partir del obtenido en la sección "2 Modelo y control tridimensional de la plataforma multirrotor". A continuación, implementaremos un controlador para el manipulador también de forma aislada, como si la base fuera fija, y finalmente se tratará de controlar el sistema acoplado. Para este último caso, a excepción de ciertos ajustes en las constantes de los controladores desarrollados previamente, utilizaremos la misma implementación que por separado. De esa forma, los controladores (del UAV y del manipulador) estarán desacoplados y actuarán de forma independiente manifestándose sus actuaciones como perturbaciones sobre el otro, que deberá ser capaz de rechazarlas.

Se sabe que una actuación, por ejemplo, en el primer motor del brazo por parte del controlador del manipulador afectará también a la plataforma, puesto que en la articulación existen fuerzas de reacción vincular. También están acopladas las inercias del sistema, de forma que, si la plataforma se mueve de forma brusca hacia la derecha, la tendencia inicial del brazo es la de quedarse en su posición inicial, aunque luego sea arrastrado. Todos estos efectos se tratarán desde el punto de vista de los controladores como perturbaciones externas que, como se dijo en el párrafo anterior, aparecen de forma natural gracias al modelo conjunto obtenido al final de la sección anterior.

4.1 Plataforma en 2D

4.1.1 Control

Para el control de la plataforma en 2D, vamos a utilizar exactamente la misma estructura que se desarrolló en el epígrafe "2.2 Control" particularizando el caso tridimensional según las simplificaciones vistas en "3.1.2 Obtención mediante particularización de las ecuaciones 3D". Partimos desde el modelo creado para el caso 3D y lo simplificamos eliminando todas aquellas variables que no involucren un movimiento estrictamente dentro del plano vertical. También se cambian las ecuaciones del controlador y del modelo a las obtenidas mediante las particularizadiones antes mencionadas. El resultado puede verse en el anexo D.

Por supuesto, la importante simplificación que nos permitió linealizar el sistema, "Ecuación 2.9", en la que declarábamos que el lazo de control del *yaw* debía ser mucho más rápido que el bucle de traslación ya no existe, puesto que la orientación es realmente constante.

4.1.2 Simulación

Se realiza una simulación simple del sistema, con unas referencias de entrada dadas como escalones. Al tratarse de una particularización exacta del modelo en el caso 3D, los resultados obtenidos serán iguales, pero teniendo únicamente las variables correspondientes al plano. Las trayectorias simuladas, para su comparación con los obtenidos en "2.3 Simulación", se muestran a continuación.

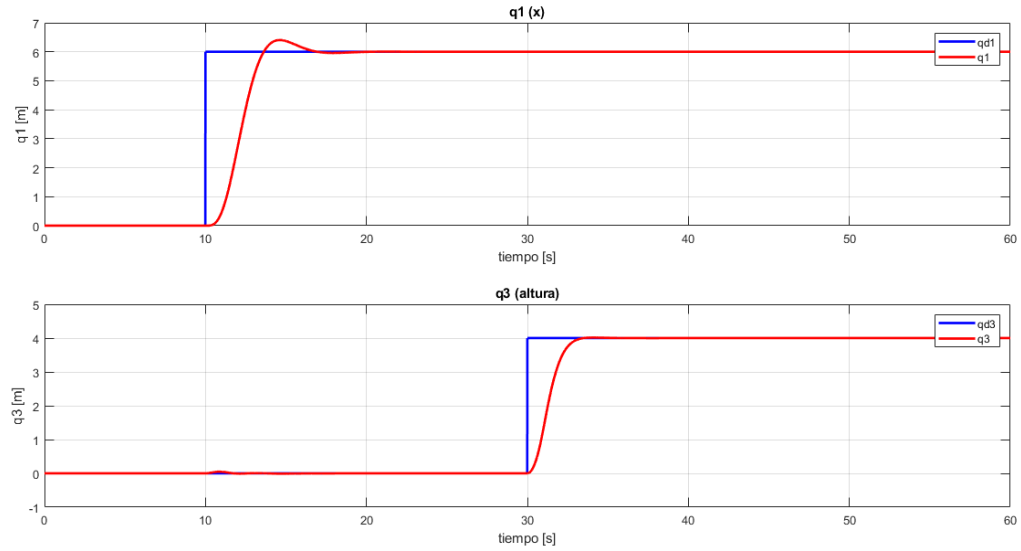


Figura 4.1 Simulación en 2D – 1

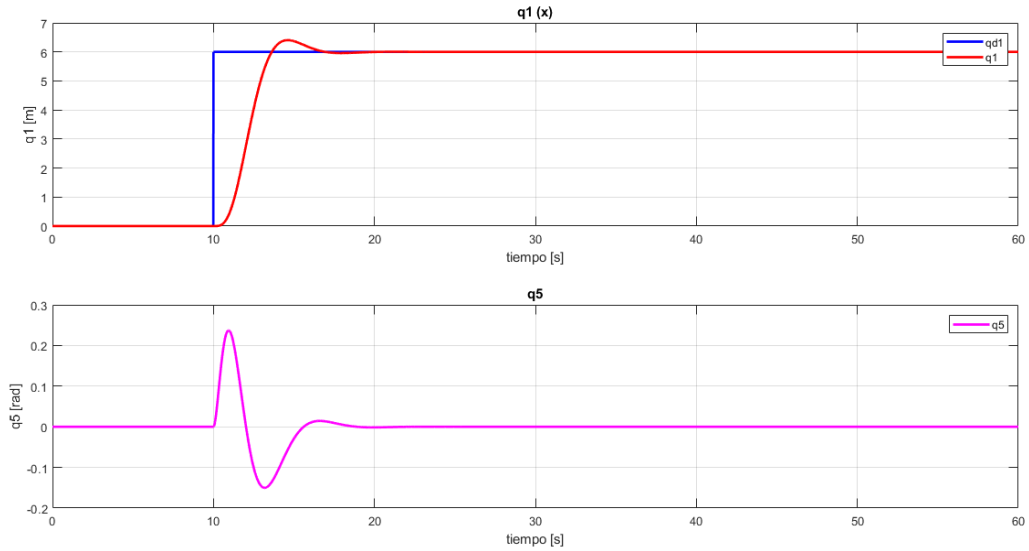


Figura 4.2 Simulación 2D – 2

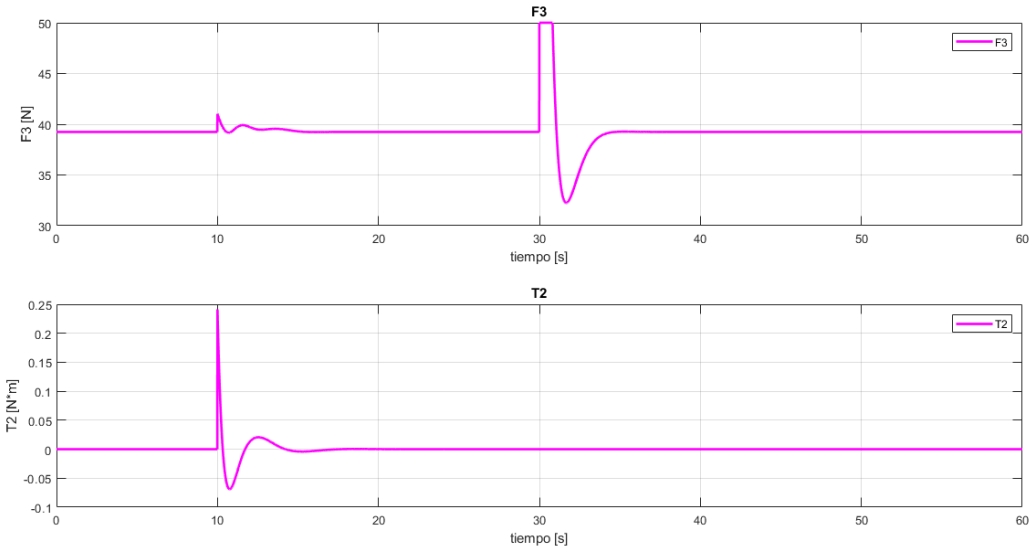


Figura 4.3 Simulación 2D – 3

Como esperábamos, los resultados son exactamente iguales que los del caso tridimensional, a excepción de la desaparición de las pequeñas perturbaciones en q_1 y q_3 que provocaban el movimiento a lo largo del eje "y" (q_2) y el giro correspondiente al *yaw*.

4.2 Brazo manipulador

4.2.1 Control

Para la implementación del controlador del brazo manipulador se ha elegido seguir el desarrollo del capítulo 10 de [8], que es la continuación directa del modelado del manipulador que realizamos en "3.2 Modelo del brazo manipulador" para el control de manipuladores no lineales.

En ese desarrollo, se detallan los pasos necesarios para obtener una implementación de un controlador por "Par Computado" o "Par Calculado". Es un método que se basa en la utilización de las matrices M, V y G obtenidas del modelado dinámico en forma cerrada para el cálculo numérico del par a aplicar en cada uno de los dos motores de las articulaciones.

La estructura de control a seguir es la siguiente:

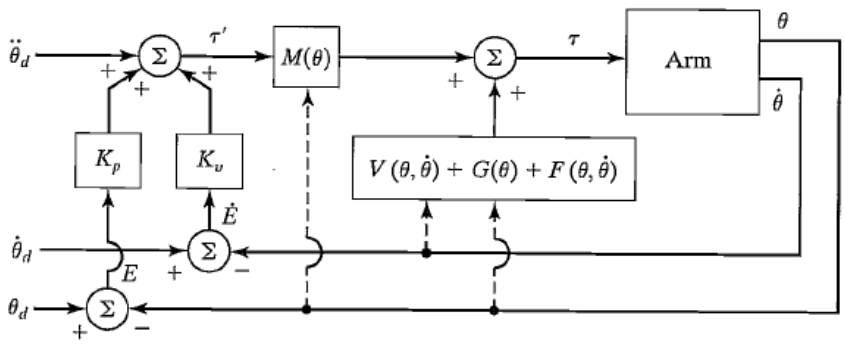


Figura 4.4 Estructura de control por par calculado

Fuente: [8]

El hecho de sumar la matriz G, que contiene los efectos de la gravedad, proporciona unos resultados parecidos a los que se obtendrían mediante un control de tipo "PD + precompensador" (estructura mediante la cual se eliminan en gran medida los problemas de error en régimen permanente sin necesidad de efecto integrador), aunque el controlador implementado sí tiene en cuenta la dinámica del manipulador.

La "Ecuación 3.26" puede verse de la forma:

$$\tau = \alpha\tau^* + \beta \qquad \textit{Ecuación 4.1}$$

Donde τ es el vector 2x1 con los pares de las dos articulaciones, τ* es el par deseado, y α y β se definen como:

$$\begin{cases} \alpha = M(q) \\ \beta = V(q,\dot{q}) + G(q) \end{cases} \qquad \textit{Ecuación 4.2}$$

Esta es la linealización que vamos a seguir en nuestro desarrollo, aunque a menudo en los robots manipuladores industriales puede observarse un control en el que la linealización es absoluta, con $\alpha = I$ y $\beta = 0$. Otro asunto que debemos explicar es el hecho de que en la realidad es muy difícil conocer a la perfección los parámetros del manipulador. Por muy bien que se hiciera la determinación de estos, con el paso de los años la degradación del propio robot haría que cambiaran, así como lo hacen en ciertas situaciones de operación como cuando el robot está soteniendo una carga con su efector. Esta imposibilidad de tener un modelo perfecto del brazo implica una mayor dificultad a la hora de analizar el bucle de control, puesto que al existir discrepancias entre el modelo computado con el sistema real puede existir error en régimen permanente. Dicho error puede ser solucionado mediante un controlador con acción integral; en nuestro caso, además de la linealización mediante las matrices M, V y G, tendremos un controlador *joint-by-joint* con acción proporcional, derivativa e integral (PID).

A continuación, se presenta la estructura de control implementada, que difiere de la mostrada en la "Figura 4.4 Estructura de control por par calculado" únicamente en que se ha añadido la acción integral, como decíamos.

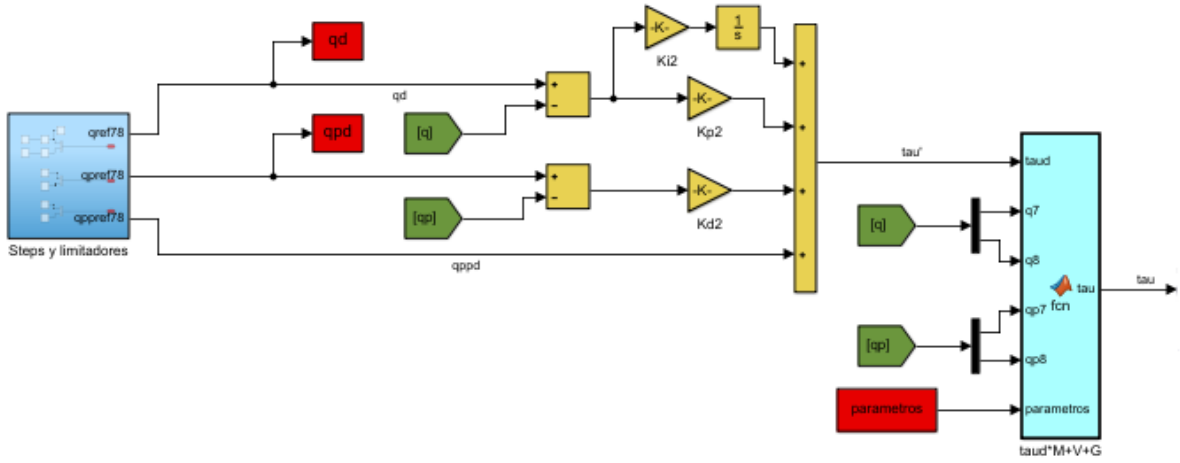


Figura 4.5 Estructura del control del manipulador

Fuente: elaboración propia

El primer subsistema, llamado "Steps y limitadores" correspondería a la etapa de generación de trayectorias, que para esta simulación no es más que un escalón en la referencia de posición y valores constantes e iguales a cero en la referencia de velocidad y aceleración angular (sin consigna). Además, al *step* de la posición angular se le añade un limitador de pendiente con el fin de que no se pueda dar una referencia que implique una velocidad angular mayor de un valor de 0.1 rad/s.

Para la elección de las constantes Kp, Kd y Ki del controlador, en lugar de tomar un valor inicial por el método de Ziegler-Nichols y seguir un método heurístico, se ha decidido hallar una función de transferencia aproximada mediante la linealización de las matrices M, V y G de la "Ecuación 4.1" (eliminando términos potenciales y trigonométricos). Tras aplicar la transformada de Laplace y sustituir los valores de las constantes por los de la "Tabla 3-1 Parámetros y constantes del sistema", se pueden calcular las dos funciones de transferencia $\frac{q_i}{\tau_i}(s)$, a las cuales se les halla el lugar de las raíces con la herramienta de Matlab "rltool".

Esta herramienta permite añadir y editar un controlador con las características que deseemos, a la vez que muestra el lugar de las raíces del sistema completo y la respuesta que se espera por parte del bucle ante un

escalón unitario de entrada.

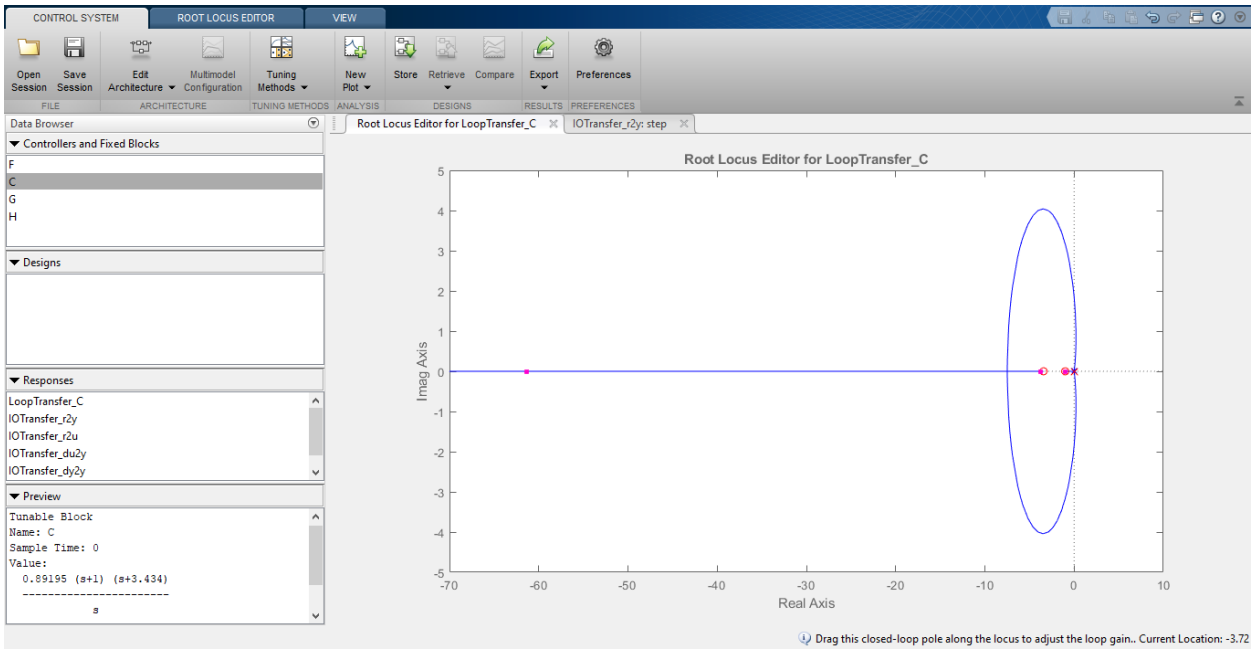


Figura 4.6 Interfaz de rltool – 1

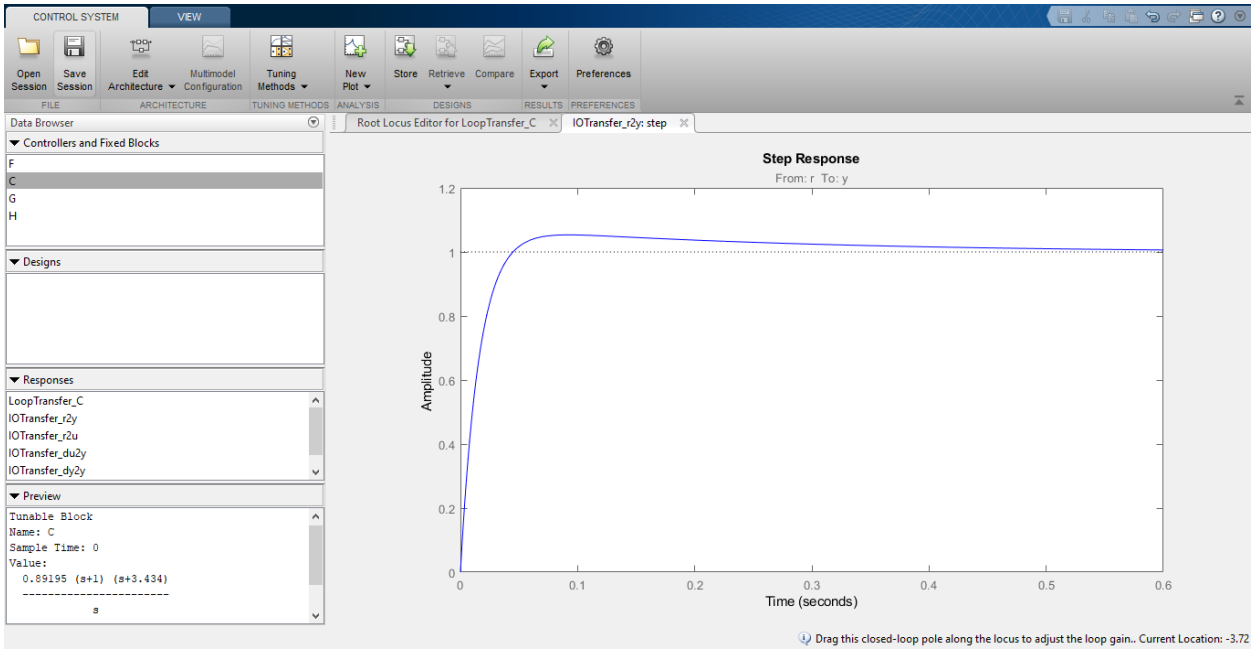


Figura 4.7 Interfaz de rltool - 2

La representación del lugar de las raíces (Figura 4.6 Interfaz de rltool – 1) permite además mover manualmente la situación de los polos y ceros del controlador. Al desear un controlador PID, debemos añadir dos ceros reales y un integrador situado en 0, puesto que la función de transferencia de estos contorladores es de la forma:

$$PID(s) = Kp\frac{Ki + s + Kd * s^2}{s}$$ ***Ecuación 4.3***

Tras unos ajustes en la situación de dichos ceros del controlador, obtenemos las siguientes constantes para los controladores:

i	Kp	Kd	Ki
1	5.108	2.333	2.000
2	2.372	1.225	0.775

Tabla 4-1 Constantes de los controladores del manipulador

El último bloque de la "Figura 4.5 Estructura del control del manipulador", es un "Matlab Fcn" de Simulink, el cual tiene como salida el vector de pares de actuación, que se calcula como $\tau = M\tau_d + V + G$, siendo M, V y G las matrices y vectores calculados en el apartado "3.2.3 Dinámica" con la consideración del epígrafe "3.3 Modelo del manipulador acoplado en la plataforma" de que G debe ir cambiada de signo. Las entradas de esta función son el par deseado, las posiciones y velocidades articulares "reales" (calculadas por el modelo) y el vector parámetros, que contiene las constantes g, L_1, L_2, m_1 y m_2.

4.2.2 Simulación

Para la simulación del sistema, nos falta implementar el modelo del brazo manipulador. Lo haremos a través de un bloque "Interpreted Matlab Fcn" de Simulink, que ejecuta en cada paso de simulación la función escrita en un fichero ".m" que se le indique. En dicha función se buscará la solución al sistema de ecuaciones del problema dinámico directo, de la forma $M\ddot{q} = \tau - (V + G)$, que no es más que un sistema matricial de la forma Ax=b y se resuelve en Matlab mediante el operador barra invertida "\".

El anexo E contiene el modelo de simulink con el controlador y el modelo del manipulador antes descritos, cuya simulación proporciona los siguientes gráficos:

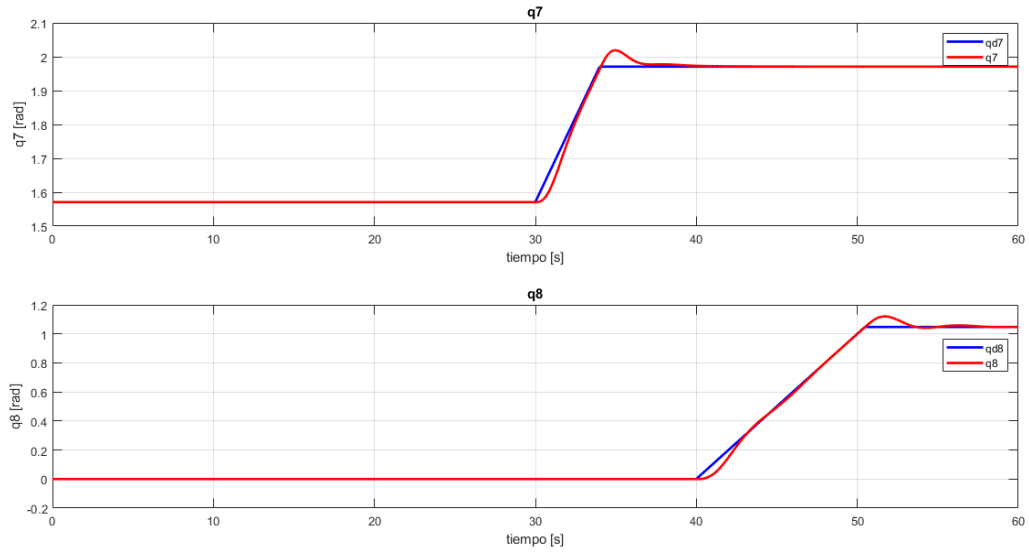


Figura 4.8 Simulación solo manipulador

El seguimiento de las trayectorias del manipulador es satisfactorio, presentando una respuesta temporal parecida a la que se ha observado por parte de la plataforma en apartados anteriores.

4.3 Manipulador aéreo

En este epígrafe se implementa conjuntamente todo lo desarrollado hasta ahora. Por el momento se ha desarrollado un modelo para la plataforma aislada, su controlador correspondiente y hemos realizado simulaciones. De igual forma hemos tomado el manipulador aislado, lo hemos modelado, se ha implementado un controlador y se ha simulado también. Por último, obtuvimos por medio del programa Autolev las ecuaciones de la cinemática y dinámica que modelan al sistema completo UAV más manipulador acoplado.

4.3.1 Control

Como ya se mencionó anteriormente, se va a realizar el control de la plataforma y del manipulador de forma independiente, como si cada uno de estos dos subsistemas no estuvieran acoplados. Así, cuando el control de traslación de la base voladora actúe, se provocarán perturbaciones sobre el manipualdor que su controlador deberá compensar, y viceversa. Este hecho, por supuesto, tenderá a desestabilizar el sistema completo, provocando grandes oscilaciones si los controladores no pueden rechazar dichos efectos, y llevando el UAV a un comportamiento inestable en el peor de los casos.

La implementación del sistema completo es la siguiente.

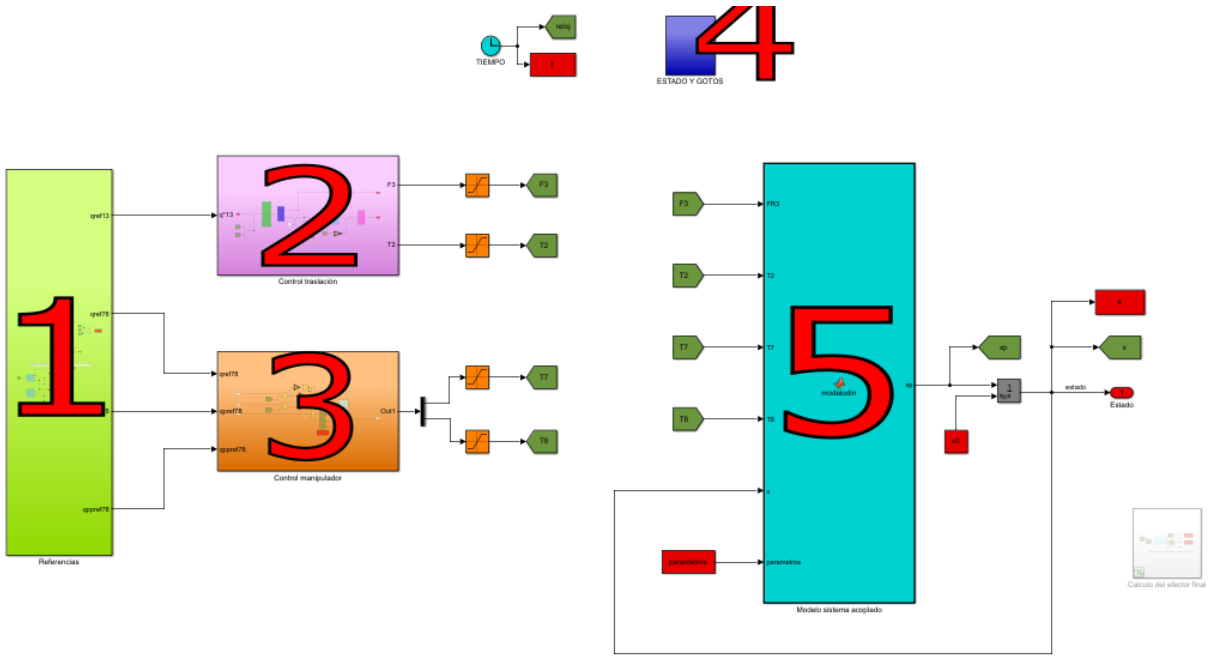


Figura 4.9 Control y sistema UAV manipulador

Fuente: elaboración propia

Comenzamos por el subsistema marcado con 1 en la figura anterior, que correspende a la generación de señales de referencia de q_1, q_3, q_7 y q_8. Como vimos en el apartado de control del manipulador, podemos establecer también referencias para la velocidad y aceleración angular del brazo, pero en nuestro caso valdrán cero. El interior de dicho bloque se puede ver a continuación.

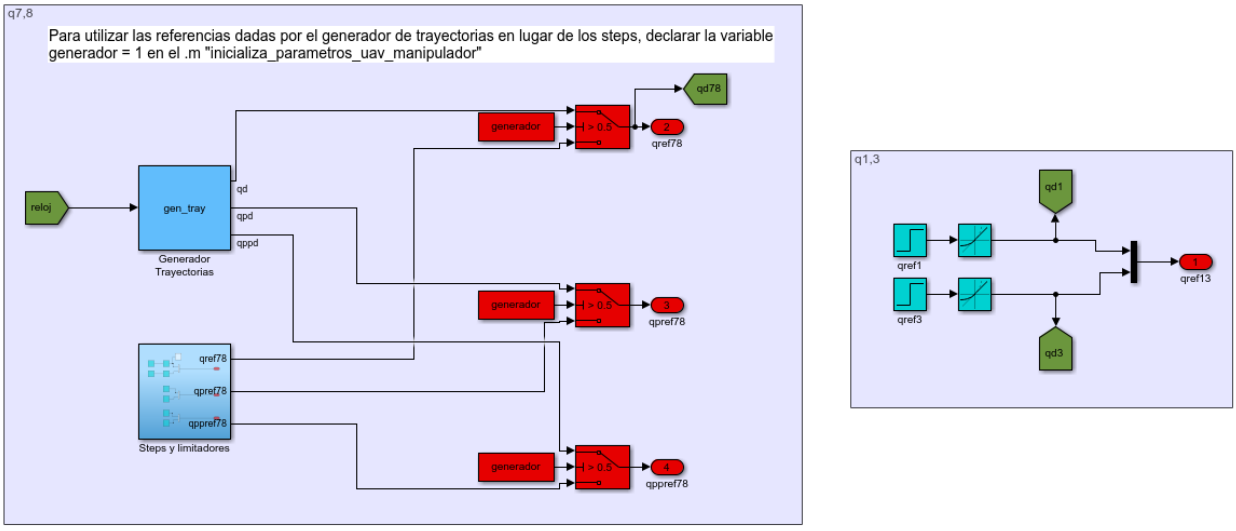


Figura 4.10 Generación de trayectorias de referencia

Fuente: elaboración propia

Dentro del subsistema "steps y limitadores" se encuentran las referencias en escalón más unos limitadores de pendiente para evitar picos de aceleraciones. En el caso de las posiciones lineales q_1 y q_3 esta pendiente está limitada a una velocidad máxima de 0.3 m/s, y para las angulares q_7 y q_8, de 0.1 rad/s (y el mismo valor, pero en negativo para bajadas). En una extensión del problema, estas referencias provendrían de una etapa de planificación, pero este contenido queda fuera de los límites que abarca este TFG.

En el subsistema marcado con un 2 en la "Figura 4.9 Control y sistema UAV manipulador" se implementa el controlador de la plataforma. Este controlador excepto por las constantes del PID es exactamente el mismo que en el apartado "4.1.1 Control", y puede verse a continuación.

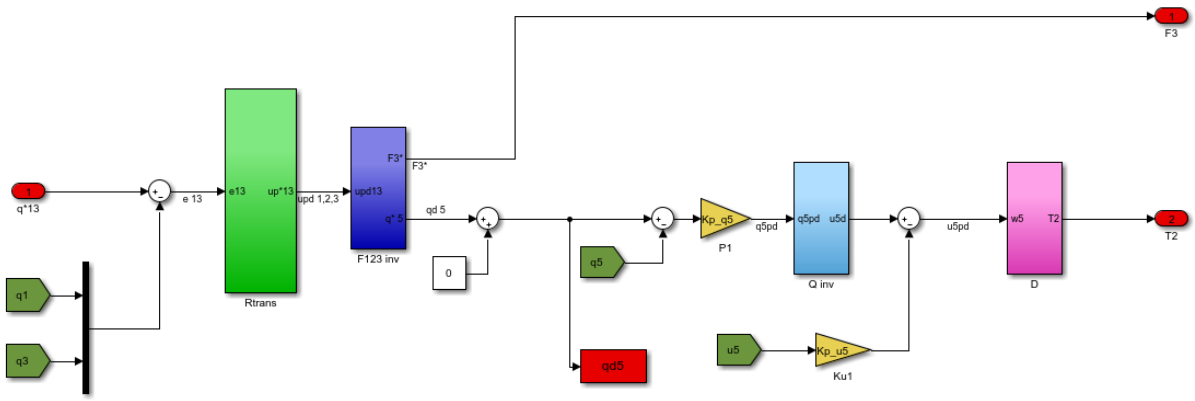


Figura 4.11 Controlador de traslación del UAV

Fuente: elaboración propia

A falta de verificación mediante experimentación para comprobar que no se exceden las limitaciones físicas del sistema, se han situado los polos de las funciones de transferencia del control de traslación en -4 en lugar de -1. Las funciones de transferencia son las presentadas en "2.2.3 Asignación de polos", y recordamos que los polos se situaron en -1 por la recomendación explícita de [2]. También deberíamos recordar que -1 era una buena localización para los polos del bucle de traslación porque a parte necesitábamos un segundo bucle de control para el *yaw*, que debía ser mucho más rápido que el primero para poder linealizar el sistema. Al no existir un segundo bucle de control de orientación en el caso 2D, no existe inconveniente con el desplazamiento de los polos.

No ha sido el caso del controlador del manipulador, que podemos ver en la "Figura 4.9 Control y sistema UAV manipulador" marcado con un 3. Su implementación ha sido exactamente igual que en el punto "4.2 Brazo manipulador", incluyendo los parámetros de la "Tabla 4-1 Constantes de los controladores del manipulador".

4.3.2 Simulación

El subsistema 4 de la "Figura 4.9 Control y sistema UAV manipulador" no es parte ni del control ni del modelo del sistema, sino de la implementación en simulink. Contiene varias etiquetas "goto" y "from" usadas para aliviar el cableado en el programa, así como bloques "to workspace", para guardar datos en variables y poder representarlas posteriormente. Su contenido puede verse a continuación.

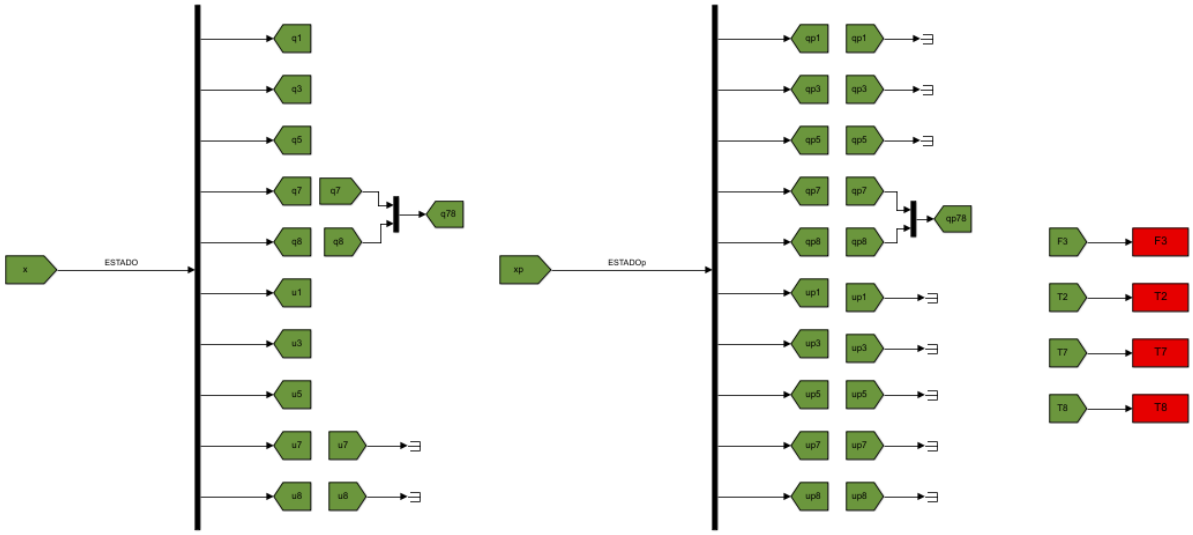


Figura 4.12 Contenido del bloque de "goto's"

Fuente: elaboración propia

Por último, queda implementar el subsistema marcado con un 5, que es un bloque "Matlab Fcn" con el modelo mecánico completo del manipulador aéreo. Recordamos que estas ecuaciones las obtuvimos mediante el programa Autolev, al que le dimos la orden de generar el fichero con la dinámica resuelta (fichero "dynamics_coded.m" mencionado en "3.3 Modelo del manipulador acoplado en la plataforma") para la implementación en Matlab. El código autogenerado no tiene el mismo formato que el que necesitamos en el modelo de Simulink, y además no todo su contenido nos interesa (únicamente nos quedamos con la sección en la que se resuelve el sistema de ecuaciones que da la solución a las aceleraciones $\dot{u}_i$). A continuación, se muestra el código de dicho bloque:

```
function xp = modelodin(FR3,T2,T7,T8,x,parametros)

xp  = zeros(10,1);
COEF = zeros(5,5);
RHS = zeros(1,5);

%ENTRADAS

q1=x(1); q3=x(2); q5=x(3); q7=x(4);q8=x(5);
u1=x(6); u3=x(7); u5=x(8); u7=x(9); u8=x(10);

M=parametros(1);
g=parametros(2);
I11=parametros(3);
I22=parametros(4);
I33=parametros(5);
l1  = parametros(6);
l2  = parametros(7);
m1  = parametros(8);
m2  = parametros(9);
d   = parametros(10);

% Ecuaciones cinemáticas:
q1p = u1;
q3p = u3;
q5p = u5;
q7p = u7;
q8p = u8;

% Ecuaciones dinámicas:
```

```
COEF(1,1) = -M - m1 - m2;
COEF(1,2) = 0;
COEF(1,3) = m1*(d*cos(q5)+l1*sin(q5+q7)) +
m2*(d*cos(q5)+l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(1,4) = l1*m1*sin(q5+q7) + m2*(l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(1,5) = l2*m2*sin(q5+q7+q8);
COEF(2,1) = 0;
COEF(2,2) = -M - m1 - m2;
COEF(2,3) = -m1*(d*sin(q5)-l1*cos(q5+q7)) - m2*(d*sin(q5)-l1*cos(q5+q7)-
l2*cos(q5+q7+q8));
COEF(2,4) = l1*m1*cos(q5+q7) + m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8));
COEF(2,5) = l2*m2*cos(q5+q7+q8);
COEF(3,1) = m1*(d*cos(q5)+l1*sin(q5+q7)) +
m2*(d*cos(q5)+l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(3,2) = -m1*(d*sin(q5)-l1*cos(q5+q7)) - m2*(d*sin(q5)-l1*cos(q5+q7)-
l2*cos(q5+q7+q8));
COEF(3,3) = -I22 - m1*(d^2+l1^2+2*d*l1*sin(q7)) -
m2*(d^2+l1^2+l2^2+2*d*l1*sin(q7)+2*l1*l2*cos(q8)+2*d*l2*sin(q7+q8));
COEF(3,4) = -l1*m1*(l1+d*sin(q7)) -
m2*(l1^2+l2^2+d*l1*sin(q7)+2*l1*l2*cos(q8)+d*l2*sin(q7+q8));
COEF(3,5) = -l2*m2*(l2+l1*cos(q8)+d*sin(q7+q8));
COEF(4,1) = l1*m1*sin(q5+q7) + m2*(l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(4,2) = l1*m1*cos(q5+q7) + m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8));
COEF(4,3) = -l1*m1*(l1+d*sin(q7)) -
m2*(l1^2+l2^2+d*l1*sin(q7)+2*l1*l2*cos(q8)+d*l2*sin(q7+q8));
COEF(4,4) = -m1*l1^2 - m2*(l1^2+l2^2+2*l1*l2*cos(q8));
COEF(4,5) = -l2*m2*(l2+l1*cos(q8));
COEF(5,1) = l2*m2*sin(q5+q7+q8);
COEF(5,2) = l2*m2*cos(q5+q7+q8);
COEF(5,3) = -l2*m2*(l2+l1*cos(q8)+d*sin(q7+q8));
COEF(5,4) = -l2*m2*(l2+l1*cos(q8));
COEF(5,5) = -m2*l2^2;
RHS(1) = m1*(d*sin(q5)*u5^2-l1*cos(q5+q7)*(u5+u7)^2) + m2*(d*sin(q5)*u5^2-
l1*cos(q5+q7)*(u5+u7)^2-l2*cos(q5+q7+q8)*(u5+u7+u8)^2) - FR3*sin(q5);
RHS(2) = g*M + g*m1 + g*m2 + m1*(d*cos(q5)*u5^2+l1*sin(q5+q7)*(u5+u7)^2) +
m2*(d*cos(q5)*u5^2+l1*sin(q5+q7)*(u5+u7)^2+l2*sin(q5+q7+q8)*(u5+u7+u8)^2) -
FR3*cos(q5);
RHS(3) = g*m1*(d*sin(q5)-l1*cos(q5+q7)) + g*m2*(d*sin(q5)-l1*cos(q5+q7)-
l2*cos(q5+q7+q8)) - T2 - d*l1*m1*cos(q7)*(u5^2-(u5+u7)^2) -
m2*(d*l1*cos(q7)*u5^2+d*l2*cos(q7+q8)*u5^2+l1*l2*sin(q8)*(u5+u7+u8)^2-
d*l1*cos(q7)*(u5+u7)^2-l1*l2*sin(q8)*(u5+u7)^2-
d*l2*cos(q7+q8)*(u5+u7+u8)^2);
RHS(4) = m2*(l1*l2*sin(q8)*(u5+u7)^2-d*l1*cos(q7)*u5^2-d*l2*cos(q7+q8)*u5^2-
l1*l2*sin(q8)*(u5+u7+u8)^2) - T7 - g*l1*m1*cos(q5+q7) -
g*m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8)) - d*l1*m1*cos(q7)*u5^2;
RHS(5) = -T8 - g*l2*m2*cos(q5+q7+q8) - l2*m2*(d*cos(q7+q8)*u5^2-
l1*sin(q8)*(u5+u7)^2);
SolutionToAxEqualsB = COEF\RHS';

% Update variables after uncoupling equations
u1p = SolutionToAxEqualsB(1);
u3p = SolutionToAxEqualsB(2);
u5p = SolutionToAxEqualsB(3);
u7p = SolutionToAxEqualsB(4);
u8p = SolutionToAxEqualsB(5);

% Output:
xp(1)  = q1p;
xp(2)  = q3p;
xp(3)  = q5p;
xp(4)  = q7p;
```

```
xp(5)  = q8p;
xp(6)  = u1p;
xp(7)  = u3p;
xp(8)  = u5p;
xp(9)  = u7p;
xp(10) = u8p;
```

En el anexo F se muestran tanto el modelo de simulink como el fichero de código .m en el que se da valor a todos los parámetros involucrados tanto en la construcción física del sistema como en los controladores. También en ese .m se establecen los tiempos de simulación en los que se introducen las referencias de las entradas en escalón y los valores de estos. Simulando el sistema podemos observar su comportamiento:

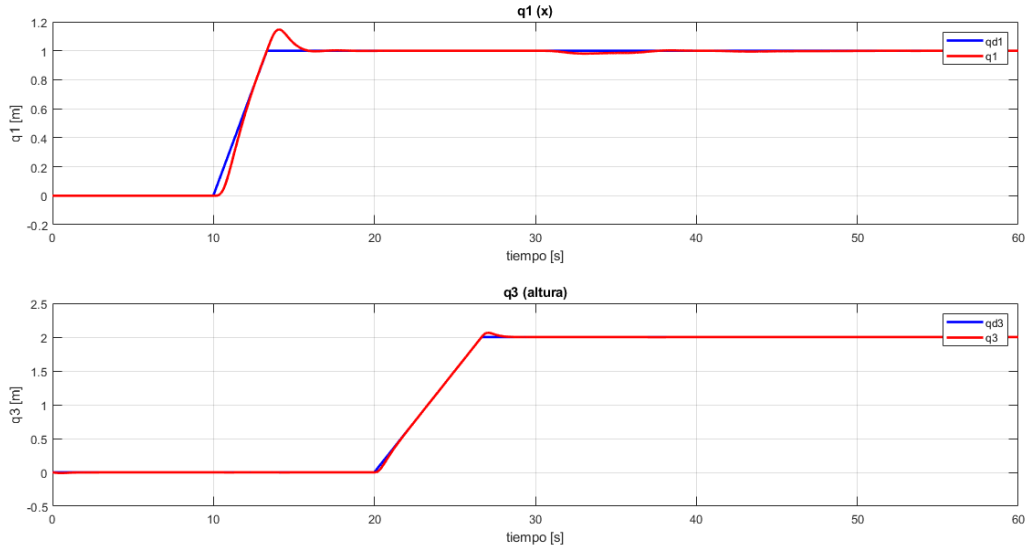


Figura 4.13 Simulación UAV manipulador – 1

La reubicación de los polos del control de traslación ha servido para controlar finalmente la posición del centro de gravedad de la plataforma.

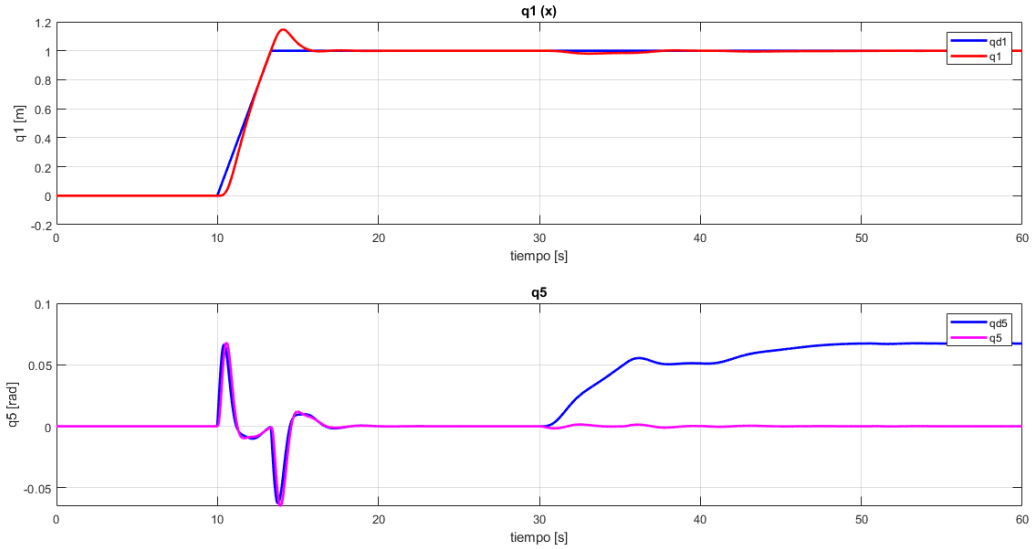


Figura 4.14 Simulación UAV manipulador - 2

En la anterior gráfica se observan las curvas de q_1 y del giro q_5. Como sabemos, el giro según q_5 es el que permite los desplazamientos horizontales del vehículo (según el eje n_1). Entre los segundos 10 y 15 de simulación

podemos observar un comportamiento normal de esta variable, girando la plataforma para moverla hacia delante (y hacia atrás en la sobreoscilación). El comportamiento a partir del segundo 30 puede llamar la atención, dado que el giro q_5 real no coincide con el que el controlador establece como el deseado. Esto se debe a que en ese instante (segundo 30) se produce un giro del brazo manipulador que inicialmente estaba completamente recto y hacia abajo, lo que hace que la plataforma tienda a girar y necesite compensarlo. Esta actuación se realiza mediante T_2, que está directamente relacionada con q_5, y la única forma que tiene el controlador de provocar un par es creando una q5d no nula. Este efecto puede verse representado con más detalle en las siguientes gráficas; los esfuerzos sobre la plataforma:

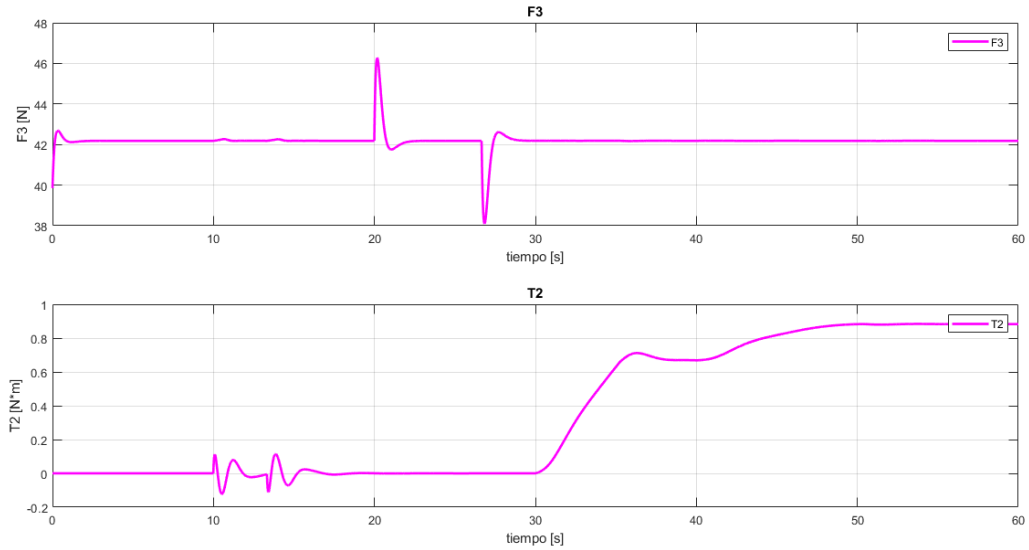


Figura 4.15 Simulación UAV manipulador - 3

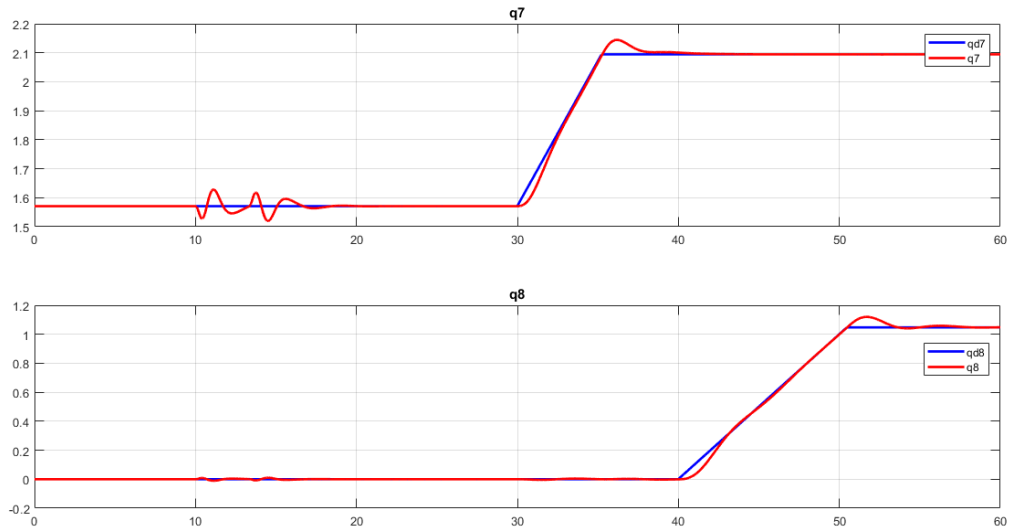


Figura 4.16 Simulación UAV manipulador - 4

En la anterior gráfica se ilustran las trayectorias seguidas por las articulaciones. El comportamiento es bastante parecido al obtenido cuando lo simulamos sin acoplarlo en la plataforma, ya que los parámetros del controlador son los mismos. Podemos observar en el segundo 10 la perturbación que aparece sobre q_7 y q_8 al mover la plataforma horizontalmente, que hace oscilar levemente el brazo extendido. Por último, podemos ver los pares aplicados en cada una de las articulaciones del manipulador junto a las trayectorias articulares descritas:

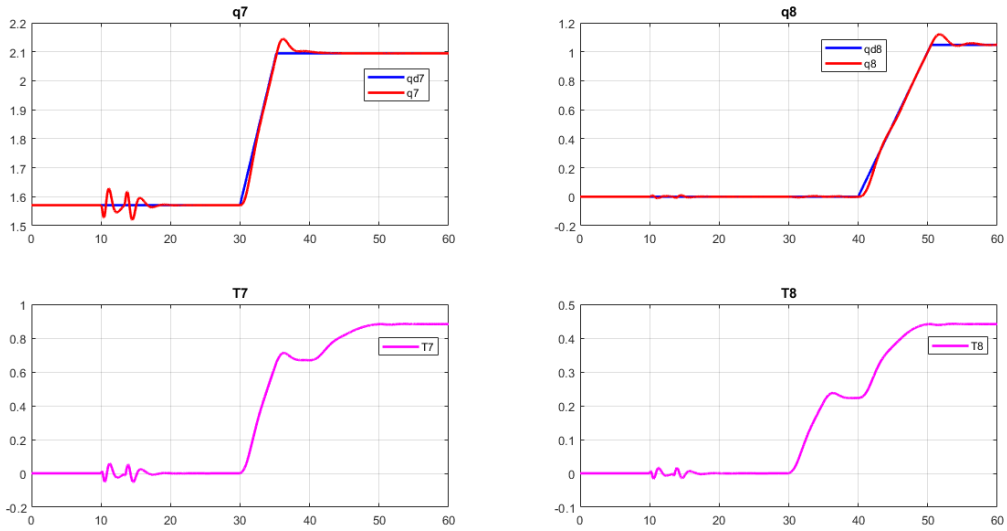


Figura 4.17 Simulación UAV manipulador – 5

5 MODELADO DE LA SENSORIZACIÓN

La sensorización es el siguiente problema que abordar en nuestra escala hacia un modelo lo más cercano a la realidad posible. Atañe a cualquier sistema real que necesite un control en bucle cerrado, puesto que los sensores son los elementos que proporcionan la información del estado del sistema en cada momento. La única solución que no involucra sensores es el control en bucle abierto, opción que queda de inmediato descartada puesto que necesitamos observar un estado para cumplir con los requisitos de la tarea.

La odometría es la técnica por medio de la cual un sistema móvil, como nuestro UAV, trata de estimar su posición por medio de la lectura e integración de las medidas de diversos sensores a lo largo del tiempo. Existen una infinidad de sensores que miden todo tipo de magnitudes, de cualquier rango de precios y muchísimos tipos de montaje, pero para la estimación de posición de robots móviles tenemos un grupo reducido de sensores que se utilizan con mucha frecuencia. El GPS, el acelerómetro, el giróscopo y el magnetómetro son las soluciones más extendidas para este problema.

Algunos sensores proporcionan medidas de posición absolutas que no van acumulando error a lo largo de la trayectoria, permitiendo estimar la posición del sistema en todo momento. Es el caso del GPS, que cada cierta frecuencia proporciona una medida de posición; también con el acelerómetro se puede calcular mediante una relación trigonométrica simple la matriz de giro del sistema de ejes del UAV con respecto a la base fija, de forma absoluta y sin error acumulado. Otros, como el giróscopo, necesitan ser integrados en el tiempo para obtener una medida de la posición, lo cual provoca una deriva en la estimación que se hace más grande conforme se avanza; esto se debe a que, en cada paso, además de la medida se están integrando los errores de esta y el ruido de la señal del sensor.

No debemos olvidar que en este trabajo estamos modelando un manipulador aéreo, por lo que debemos tener en cuenta también los sensores que integraremos en el brazo manipulador. Al ser un manipulador únicamente con vínculos rotativos simples en las articulaciones, nuestro objetivo será conocer la posición articular de estas; para ello utilizaremos dos codificadores ópticos.

5.1 Selección de sensores

Para un UAV con movimiento restringido al plano vertical, necesitamos conocer la posición del centro de masas del vehículo con respecto al sistema de ejes fijo {n}, así como el giro de la plataforma. Para ello, haremos uso de un sistema de GPS y de una unidad de medición inercial (IMU), que es un dispositivo electrónico que integra varios sensores.

El GPS proporcionará directamente una medida de la posición absoluta del vehículo, sin error acumulado con el paso del tiempo. La desventaja del GPS es que tiene una frecuencia de muestreo baja, y que además en entornos no muy expuestos directamente al cielo presenta grandes errores de medida. En interiores ni siquiera podemos afirmar que este sistema de posicionamiento funcione. Estos puntos débiles nos obligan a instalar un método alternativo que proporcione otra estimación de la posición del vehículo para conjugarlas. A partir del modelo físico del sistema, las medidas de uno o varios sensores y el conocimiento de la incertidumbre de estos últimos, podemos obtener una estimación del estado del sistema por medio de un algoritmo como el **filtro de Kalman** (KF[3]).

El KF es un algoritmo de estimación de estado que, gracias a su implementación recursiva, puede ser utilizado en sistemas de tiempo real como el de este trabajo. Tanto la versión original como la versión extendida del filtro (EKF[4]), que es la versión para sistemas no lineales del KF, son ampliamente utilizadas en muchas ramas tecnológicas y científicas por su gran versatilidad y robustez a la hora de predecir el comportamiento de un

[3] *Kalman Filter,* Filtro de Kalman
[4] *Extended Kalman Filter,* Filtro de Kalman Extendido

sistema.

Con respecto al resto de sensores que implementaremos para la medición del estado, tendremos también una IMU compuesta por acelerómetro y giróscopo. El primero de ellos proporciona en cada ciclo una medida de la aceleración sufrida a lo largo de cada uno de los tres ejes que posee. El segundo, mide en cada ciclo de muestreo la velocidad angular instantánea, también en cada uno de sus tres ejes. El acelerómetro nos permite conocer directamente la orientación del vehículo gracias a unas relaciones triginométricas sencillas. La ventaja del acelerómetro es que no acumula error a lo largo de la trayectoria, pero tiene la desventaja de que proporciona una señal muy ruidosa. Por el contrario, tenemos el giróscopo, que proporciona una medida bastante menos ruidosa que el acelerómetro, pero acumula error a lo largo de la trayectoria. Esto se debe a que el giróscopo mide la velocidad angular vista en cada uno de sus tres ejes, la cual debemos integrar para conocer posición absoluta. Estos dos sensores de la IMU se complementan muy bien y son muy utilizados para odometría, por lo que ya existen soluciones que tratan de tomar lo mejor de cada uno de ellos y usarlos en conjunto para que cada uno compense las desventajas del otro. Por ejemplo, el llamado filtro complementario utiliza las medidas del giróscopo para la estimación a corto plazo, mientras que corrige la deriva de la trayectoria con las medidas del acelerómetro, de mayor periodo. Este filtro es en sí mismo una variación del Filtro de Kalman para estado estacionario que no contempla ninguna descripción estadística de los ruidos del acelerómetro y giróscopo [12]; otra opción, más afinada, es utilizar la implementación completa del KF o EKF.

El GPS proporciona en exteriores una medida de posición muy útil, sobre todo para desplazamientos en el plano horizontal (plano del suelo). Por desgracia, no es así para el caso de movimientos verticales propios de un VTOL, para los que este sistema de posicionamiento global no es tan sensible. Motivado por esto, necesitaremos un sensor adicional que mida exclusivamente la altura de la plataforma con respecto al suelo; esto es, un altímetro. Existen a su vez varias opciones, entre ellas el altímetro barométrico, con el que se calcula la atura a través de una medida de la presión atmosférica. Otras soluciones son los ultrasonidos y los láseres, que estiman la distancia midiendo el lapso de tiempo que ocurre entre que se lanza, rebota y se recibe una onda (mecánica en el caso de los ultrasonidos y electromagnética en el caso de los láseres). En nuestro caso, elegiremos el altímetro barométrico.

El magnetómetro es otro sensor ampliamente utilizado, que otorga una estimación absoluta de la actitud[5] del sistema gracias a medidas del campo magnético terrestre y una serie de transformaciones. En un movimiento restringido al plano vertical sólo tenemos un ángulo posible de giro, en nuestro caso el que ocurre según el eje f_2, codificado por la variable q_5 (*pitch*). Ni el *yaw* ni el *roll* entrarán en juego en nuestros desarrollos, y puesto que para el único ángulo no nulo ya tenemos una medición a partir del acelerómetro, se ha decidido prescindir del magnetómetro en este desarrollo.

Por último, como mencionamos anteriormente, sólo necesitamos dos codificadores ópticos para medir la posición angular de cada una de las articulaciones del brazo manipulador. Existen varios tipos de estos sensores entre los que podemos elegir, cada uno con sus ventajas y desventajas. En un codificador óptico, el principio básico de funcionamiento consiste en observar mediante diodos fotorreceptores un disco que gira solidario al eje del motor. Dicho disco tiene orificios, de forma que cada diodo puede tener dos estados: hay luz o no la hay. Si se distribuyen correctamente estos orificios, se puede conseguir codificar cada posición del disco en un número binario que se ve reflejado en los diodos. Estas posiciones son una discretización de la circunferencia completa, de forma que podemos distinguir entre tantas posiciones como resolución tenga el codificador, cada una midiendo $\frac{360}{resolución}$ grados.

[5] Posición de giro, ángulos entre el cuerpo del UAV y el sistema de referencia global.

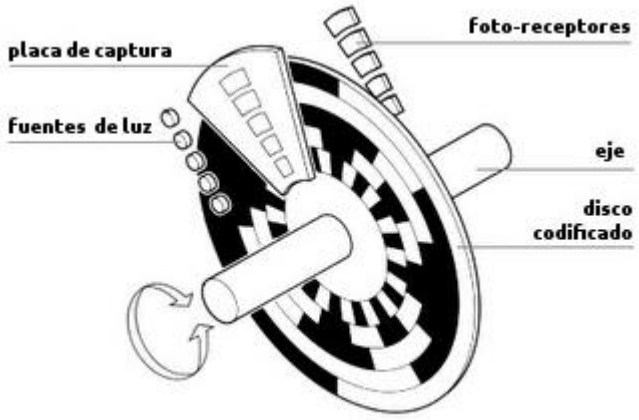


Figura 5.1 Codificador óptico absoluto

Fuente: [20]

5.2 Modelos del ruido y de los sensores

En los siguientes subapartados vamos a detallar la implementación de cada uno de los sensores. No obstante, todos ellos han sido modelados con la misma filosofía. En una aplicación real, la posición del robot es en principio desconocida, razón para la cual se equipa al UAV con distintos sensores que permitan estimar su estado completo. No ocurre así en una simulación, ya que nosotros para modelar la sensorización vamos a partir del conocimiento del estado "real" del dron, es decir, el obtenido como salida del modelo dinámico del vehículo al aplicarle las señales de actuación oportunas dictadas por el controlador. En resumen, el esquema de este apartado es el siguiente:

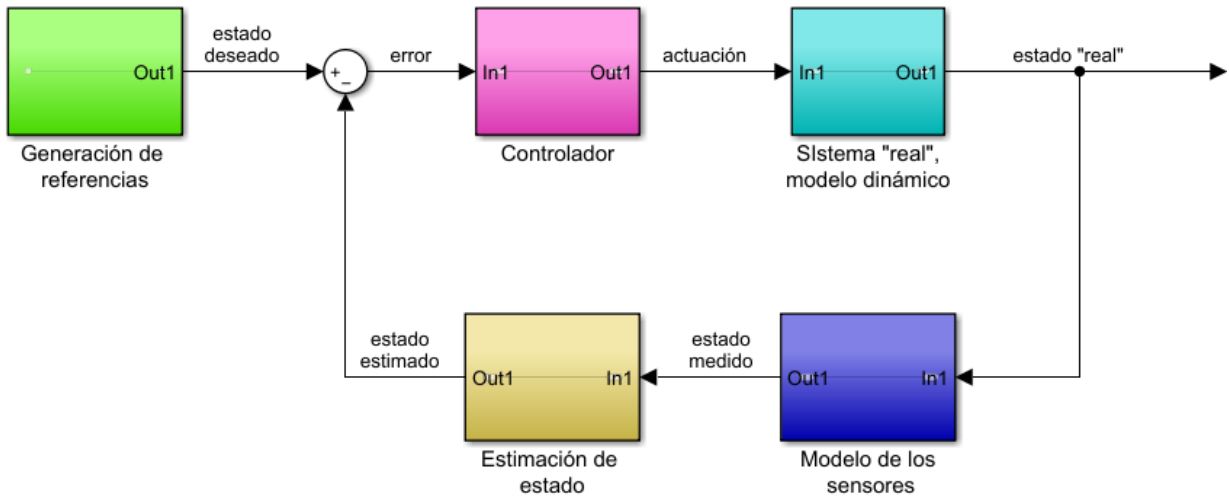


Figura 5.2 Esquema de simulación

Fuente: elaboración propia

Del estado "real" tomaremos las variables que supuestamente medirían cada sensor, o aquellas que nos permitan calcular la medida ideal esperada de cada uno de nuestros transductores para posteriormente sumarles una serie de no idealidades. Estas no idealidades son todos aquellos ruidos, desviaciones u *offsets* que llevan asociadas todas las medidas, y que en general debemos tratar de minimizar al máximo, bien adquiriendo unos sensores de mayor calidad o bien paliando sus efectos con combinaciones de sensores o algoritmos de estimación.

En cuanto al ruido de las señales, en este trabajo implantaremos dos formas distintas de modelarlo, para posteriormente hacer una comparativa entre los resultados de simulación. La primera forma será sumando a la

señal real dos componentes de ruido adicional, una de alta y otra de baja frecuencia, además de un posible *offset*. La segunda, consiste en un modelo de espacio de estados en el que una señal de ruido aleatoria pasa a través de un filtro de primer orden para ser añadida posteriormente a la ideal.

Cabe mencionar que todas las señales de ruido que vamos a contemplar serán señales de ruido gaussiano, es decir, aleatorias que siguen una distribución normal de media cero y varianza conocida: $N(\mu = 0, \sigma^2)$.

5.2.1 Modelos del ruido

5.2.1.1 Modelo del ruido de alta y baja frecuencia

Como dijimos anteriormente, de esta forma tratamos de implementar una señal ruidosa de la siguiente forma:

$$x^m = x + N(\mu = 0, \sigma^2)^{HF} + N(\mu = 0, \sigma^2)^{LF} + b \qquad \textit{Ecuación 5.1}$$

Donde x es la señal original, x^m la señal con el ruido añadido, b significa *biass* u *offset*, y los superíndices HF y LF significan alta y baja frecuenca respectivamente. Para implementar este modelo en Simulink lo hacemos de la siguiente forma:

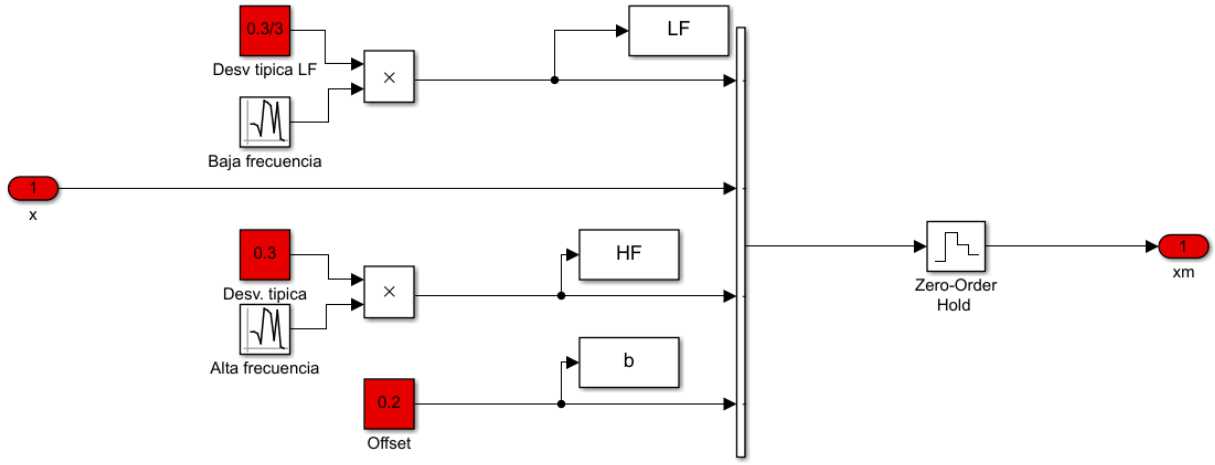


Figura 5.3 Modelo del ruido HF + LF

Fuente: elaboración propia

En la figura, x y xm son la entrada y salida del subsistema, correspondientes a las de la Ecuación 5.1, y los bloques en rojo son también entradas de datos desde el *workspace* de Matlab. Antes de la salida se añade un mantenedor de orden cero, que se encarga de mantener la salida constante durante cada periodo de muestreo. Sigma es la desviación típica del ruido del sensor, dato que estimaremos más adelante para cada sensor mediante *datasheets*[6]. Por último, las señales aleatorias se generan a través de los bloques "Random number" que hemos llamado como baja y alta frecuencia. Los parámetros que se introducen en estos bloques son:

[6] Hojas de datos. En ellas los fabricantes indican todos los parámetros y características de un dispositivo para su correcto uso.

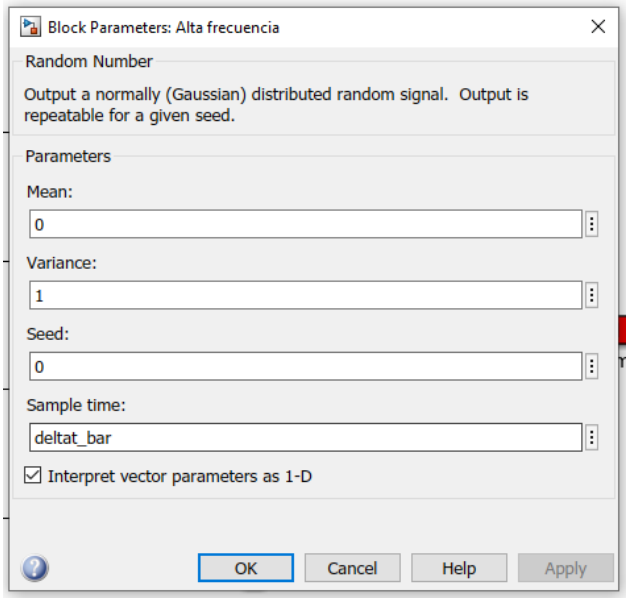


Figura 5.4 Parámetros del bloque Random Number

Fuente: elaboración propia

En ellos podemos establecer la media y varianza de una distribución normal y el tiempo de muestreo, o periodo en el que se genera cada número. En nuestro caso, en lugar de establecer directamente en el bloque la varianza σ^2, teniendo una distribución $N(\mu = 0, \sigma^2)$, estamos generando en este bloque una señal de tipo $N(0,1)$ que luego se multiplica por la desviación típica σ, siendo equivalentes ambas opciones. Por último, el tiempo de muestreo será igual a la inversa de la frecuencia de muestreo.

Si nos volvemos a fijar en la Figura 5.3 Modelo del ruido HF + LF, podemos ver que a la componente de baja frecuencia del ruido se le ha asignado una desviación típica de $\sigma/3$, al contrario que a la componente de alta, que vale σ. Además, vamos a considerar que baja frecuencia va a significar una frecuencia diez veces menor que la de alta frecuencia (o un tiempo de muestreo diez veces mayor). La alta frecuencia será para nosotros la frecuencia de muestreo de cada sensor, a la que nos proporcionan medidas. Este criterio se ha elegido porque de esta forma, con la alta frecuencia estaríamos modelando la incertidumbre en cada medida, el ruido que se detecta a simple vista al observar cualquier medición real. Por otro lado, con la baja frecuencia, modelamos unas desviaciones que duran un mayor tiempo y que pueden ocurrir en experimentos sobre todo por causas de localización o de condiciones ambiente. Por ejemplo, en el GPS puede ocurrir esto al pasar entre algunos edificios o lugares con el cielo tapado; en el magnetómetro, aunque no lo consideremos, también ocurre al pasar entre estructuras armadas con acero (ascensores, columnas, vigas, etc). Siempre podemos cambiar estos valores según la dinámica del comportamiento de los sensores en nuestra aplicación. Por ejemplo, si estamos simulando un UAV que pasa por debajo de un puente, durante el tiempo que dura ese paso a través se puede hacer que la desviación típica sea mayor que un tercio de la natural como se ha propuesto anteriormente. También se puede establecer una frecuencia menor que un décimo de la frecuencia natural para obtener perturbaciones más prolongadas en el tiempo. En resumen, se tiene una herramienta ajustable para modelar el ruido y las desviaciones que se obtienen en las medidas de un sensor.

Este modelo ha sido creado también con el objetivo de ser apto para caracterización de sistemas reales por medio de experimentación, para aplicaciones en las que no se pudiera disponer de las hojas de características de los dispositivos, por ejemplo. Mediante la caracterización de los parámetros elegidos para su diseño se podría tener una buena descripción estadística del ruido de un sensor para la mayoría de aplicaciones.

A continuación, se muestran los resultados de una simulación de este modelo de ruido, donde la señal original es constante de valor 0, la frecuencia de muestreo del sensor es de 100 Hz y la desviación típica natural de 0.3. Se supone, como en la propuesta inicial, que la componente de baja frecuencia tiene una frecuencia diez veces menor y una desviación típica de un tercio con respecto a la de alta frecuencia.

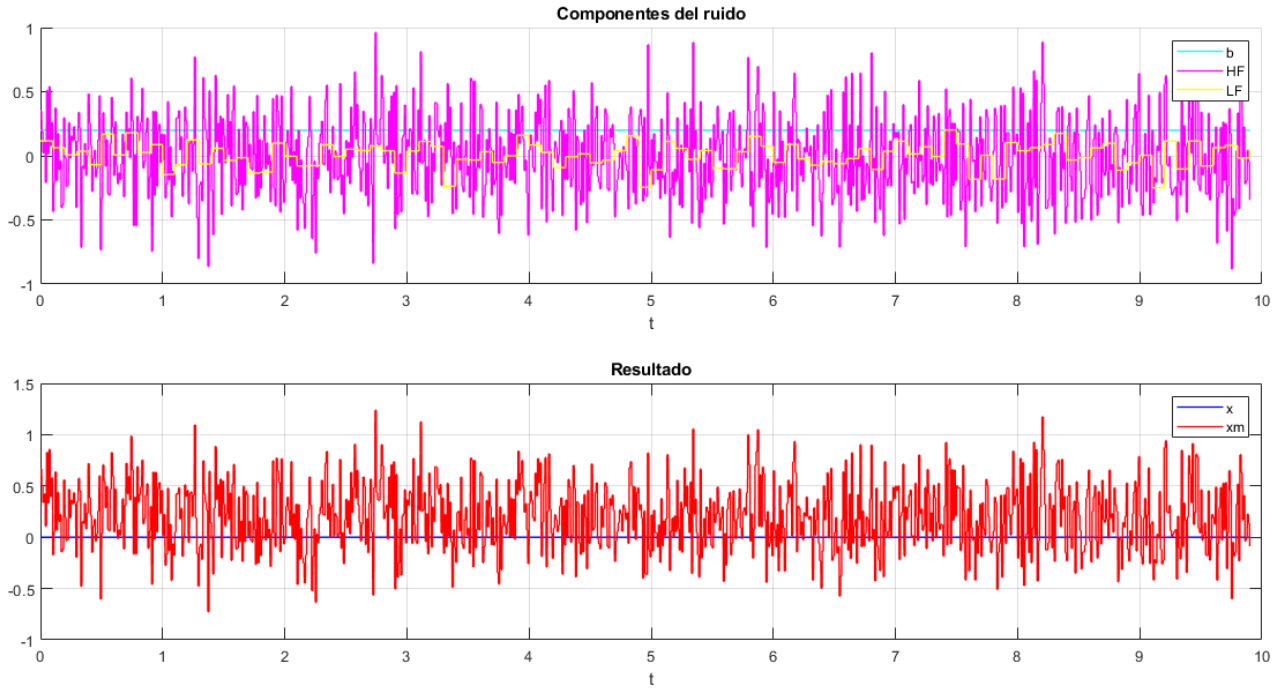


Figura 5.5 Simulación del modelo de ruido HF + LF

Fuente: elaboración propia

Haciendo un zoom a la señal medida, podemos observar con más detalle cómo es su forma:

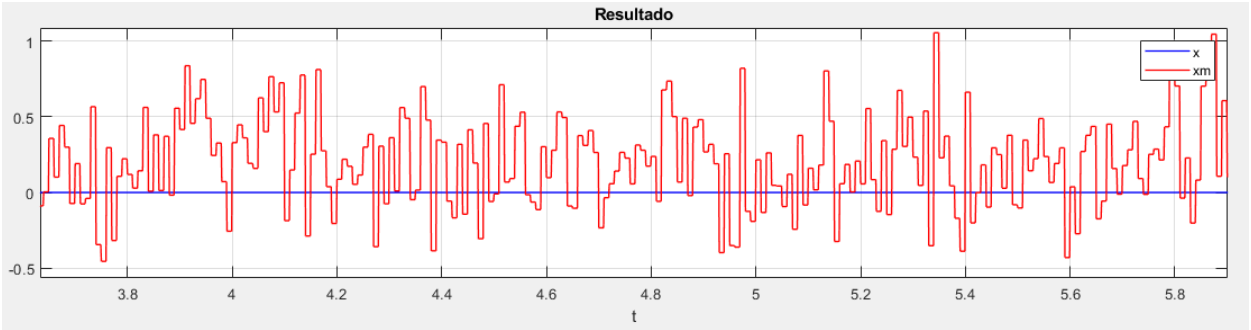


Figura 5.6 Zoom de señal medida

Fuente: elaboración propia

Apreciamos claramente el efecto del mantenedor de orden cero por la forma cuadrada de la onda.

5.2.1.2 Modelo del ruido con filtro de primer orden

A continuación, se va a desarrollar un modelo para el ruido de los sensores alternativo al creado en el epígrafe anterior. En este caso, la filosofía va a ser algo distinta: con el modelo anterior generábamos la señal de medida sumando un offset y dos componentes de ruido aleatorio a la medida ideal que se esperaría del sensor. Esto se hacía buscando siempre un modelo realista de esta onda, ya que en la realidad, aunque no se suela identificar con claridad la componente de alta o la de baja frecuencia en las hojas de características, siempre es posible hacer un análisis en frecuencia de la señal de un sensor para caracterizarlo. No obstante, en este segundo modelado del ruido obtendremos una señal de medida más continua que en el caso de HF + LF (sin un mantenedor de orden cero) y más simple, dado que únicamente haremos uso de dos parámetros para su descripción. Este modelo del ruido puede servirnos para realizar simulaciones, con el objetivo de ser comparado con el obtenido anteriormente y observar el comportamiento del UAV frente a diversas situaciones y su utilidad reside en el hecho de ser aplicado cuando se dispone de hojas de características de los sensores, puesto que los dos parámetros utilizados van muy en línea con lo que se suele aportar en ellas: frecuencia del sensor y varianza

del ruido.

La construcción de la señal medida comienza, como en el caso anterior, a partir de la medida ideal que se esperaría de un sensor sin ruido añadido ni desviaciones. A esta, sumaremos otra señal que, ahora sí, modela el ruido de la medida. Esta última onda será generada a partir de un bloque "Random number" de simulink, que como ya explicamos en su momento, proporciona datos aleatorios según una distribución normal.

Las dos únicas variables referidas al sensor en sí mismo que contemplaremos serán la frecuencia de muestreo y la desviación típica, que introduciremos en el bloque de números aleatorios exactamente igual que hicimos en el modelo de HF + LF (ver Figura 5.3 Modelo del ruido HF + LF y Figura 5.4 Parámetros del bloque Random Number). Para hacer que nuestra señal tenga una forma continua, más curveada y no cuadrada, haremos pasar la señal de números aleatorios a través de un filtro de primer orden con constante de tiempo τ. Para realizar esta operación, utilizaremos un modelo de espacio de estados que vamos a identificar a continuación.

Dicho espacio de estados se representa con las ecuaciones siguientes:

$$\begin{cases} \dot{x_k} = Ax_k + Bu_k \\ y_k = Cx_k + Du_k \end{cases} \qquad \textit{Ecuación 5.2}$$

Nuestra salida (y) será directamente el estado x, por lo que identificamos las constantes $C = 1$ y $D = 0$. Para identificar A y B, lo hacemos a partir de la función de transferencia en el dominio de Laplace del filtro de primer orden que tratamos de implementar:

$$\frac{x}{u} = \frac{1}{\tau s + 1} \qquad \textit{Ecuación 5.3}$$

Que, recolocando adecuadamente, se expresaría:

$$xs = -\frac{1}{\tau}x + \frac{1}{\tau}u \qquad \textit{Ecuación 5.4}$$

Por último, aplicando la antitransformada de Laplace, pasamos al dominio del tiempo.

$$\dot{x} = -\frac{1}{\tau}x + \frac{1}{\tau}u \qquad \textit{Ecuación 5.5}$$

De la Ecuación 5.5 podemos ya deducir las constantes $A = -\frac{1}{\tau}$ y $B = \frac{1}{\tau}$.

Por último, cabe mencionar cuál es la interpretación física de esta constante de tiempo τ. De la teoría de control, sabemos que este parámetro expresa, ante una entrada en escalón, el tiempo que tarda el sistema en alcanzar el 63% de la referencia. Es decir, que si damos una entrada en escalón unitario y $\tau = 1\ seg$, se tarda un segundo en alcanzar el valor 0.63 en la salida. Con un valor de $\tau = 4$, ya la señal alcanza el 98% de la referencia pasado un segundo, por lo que este será el valor que utilizado para diseñar.

Dado que nuestro bloque "Random number" genera una señal cuadrada que va variando su valor cada un tiempo igual a la inversa de la frecuencia del sensor, podemos interpretar estos valores como escalones de entrada para el filtro de primer orden, de forma que si ajustamos la constante de tiempo $\tau_{sensor} = 4 * \left(\frac{1}{f_{sensor}}\right)$, tendremos una onda más o menos continua que en todo momento tiende hacia el valor aleatorio generado por el bloque Random Number (el ruido). El efecto atenuante producido por el hecho de que la señal alcance el 98% de la referencia en cada periodo de muestreo es completamente despreciable frente al orden de magnitud de los valores de medidas que se tratarán.

Con esto, la implementación final de este modelo del ruido es la siguiente:

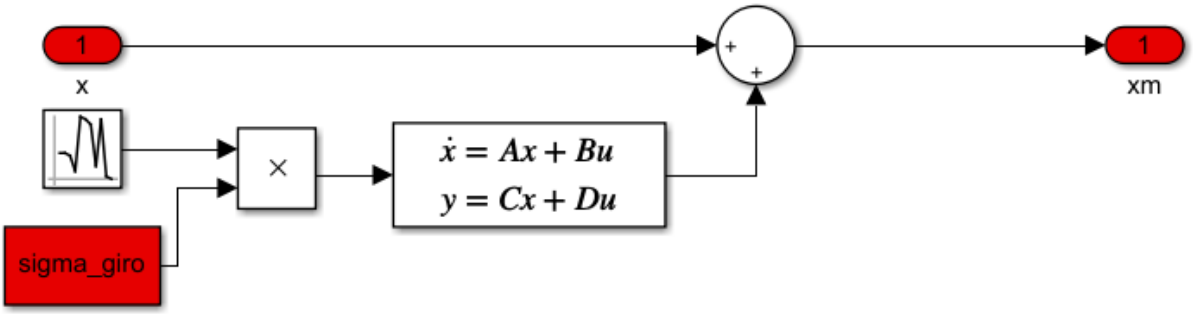


Figura 5.7 Implementación del modelo de ruido mediante espacio de estados

Fuente: elaboración propia

Los parámetros del bloque de espacio de estados son los siguientes:

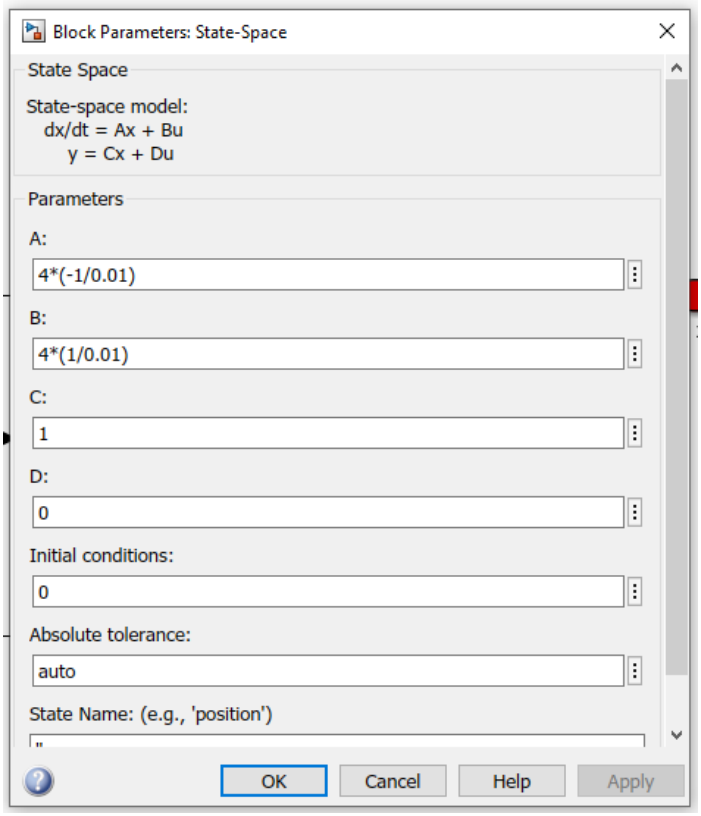


Figura 5.8 Parámetros del bloque de espacio de estados

Fuente: elaboración propia

Donde se ha tomado como ejemplo el giróscopo para ilustrar la elección de la constante de tiempo.

La señal de medida generada con este método, finalmente, tiene el siguiente comportamiento, en una simulación en la que se ha establecido como medida ideal una constante igual a 0, una frecuencia de muestreo del sensor de 100 Hz y desviación típica del ruido 1:

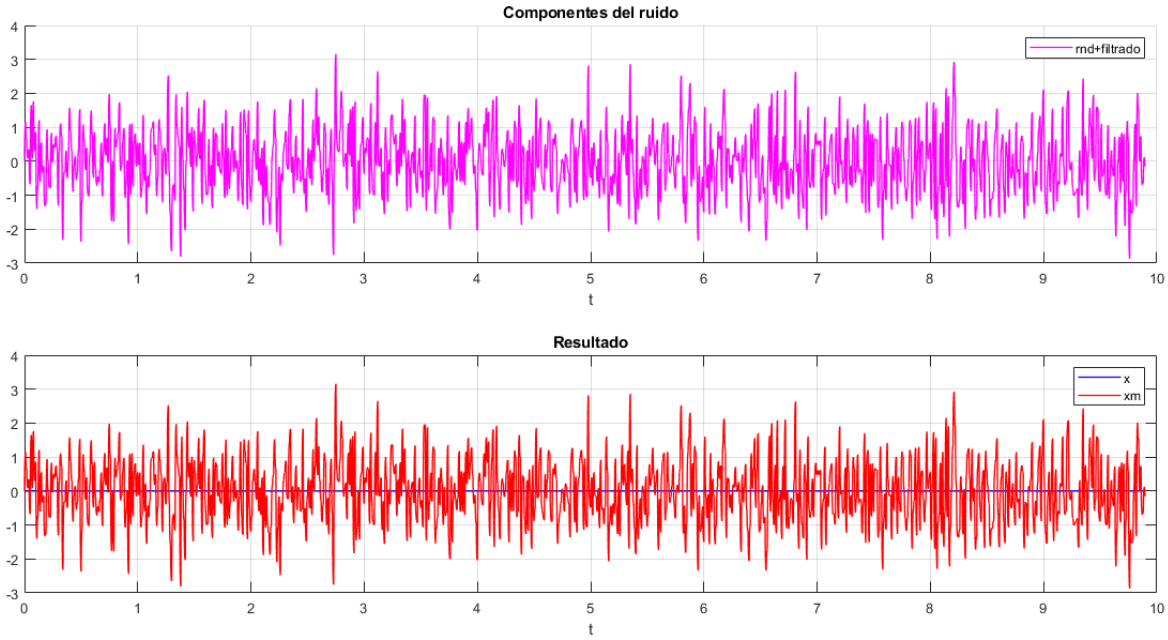


Figura 5.9 Ruido generado por el modelo de filtro de primer orden

Fuente: elaboración propia

Hacemos un zoom de la señal medida para comprobar cómo es la forma conseguida:

Figura 5.10 Zoom de la señal medida (rnd + filtr)

Fuente: elaboración propia

5.2.1.3 Comparativa.

Se realiza una simulación conjunta para realizar una comparativa entre las señales de ruido obtenidas para una medida con la misma frecuencia y desviación típica entre los dos modelos. La frecuencia elegida son 100Hz y la desviación típica 1.

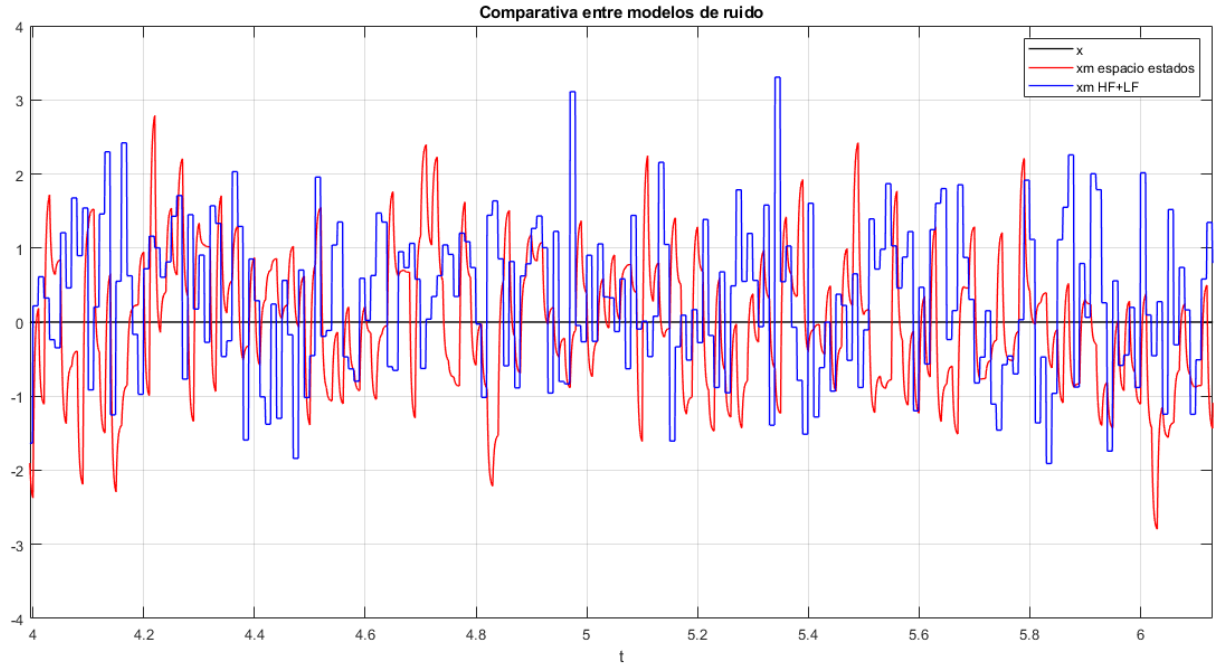


Figura 5.11 Comparativa entre el ruido generado por los modelos a igualdad de condiciones

Fuente: elaboración propia

5.2.2 Adición del ruido a la medida ideal

Como ya hemos ido explicando, la señal proporcionada por cada sensor va a ser generada mediante la suma de la medida ideal que se espera de cada sensor más el ruido generado con alguno de los dos modelos desarrollados en los apartados anteriores, siguiendo el esquema:

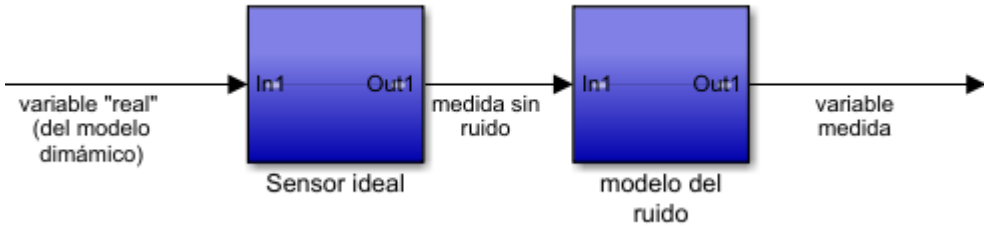


Figura 5.12 Esquema de adición de ruido a la medida

Fuente: elaboración propia

Por lo que en los siguientes epígrafes trataremos de obtener el modelo de cada uno de los sensores ideales a cuyas salidas posteriormente se les añadirá el ruido para simular las medidas.

5.2.3 Modelos de los sensores

5.2.3.1 Unidad de medición inercial

La IMU, constituida por acelerómetro y giróscopo de tres ejes cada uno.

5.2.3.1.1 Acelerómetro

El acelerómetro mide la aceleración sufrida en cada uno de sus tres ejes. Al igual que el resto de los sensores, supondremos que irá colocado solidario al UAV y en el centro de gravedad de éste. Por ello, la aceleración que estamos midiendo será la sufrida por el dron en su sistema de referencia móvil, más la ejercida por la acción de

la gravedad. Llamamos G_1 y G_3 a las medidas del acelerómetro según los ejes móviles f_1 y f_3. Para generar esta medida, haremos uso de la matriz de rotación del sistema para proyectar sobre los ejes {f} la aceleración expresada en los ejes {n}.

Sea $R_{n,f}$ la matriz de rotación que va desde {n} hasta {f}, o dicho de otra forma, la matriz de giro de {f} respecto a {n} (ya conocida y de valor $R_{n,f} = \begin{bmatrix} C_5 & 0 & S_5 \\ 0 & 1 & 0 \\ -S_5 & 0 & C_5 \end{bmatrix}$). Sabemos, del álgebra de matrices de transformación homogéneas, que para proyectar en {n} un vector expresado en las coordenadas {f} se utiliza la expresión:

$$p_n = R_{n,f} * p_f \qquad \textit{Ecuación 5.6}$$

Sabemos también que las matrices de transformación homogéneas, al ser ortonormales, cumplen lo siguiente.

$$R_{n,f} = R_{f,n}^{-1} = R_{f,n}^{T} \qquad \textit{Ecuación 5.7}$$

Por lo que, conociendo la expresión de la aceleración del UAV en los ejes {n}, $\ddot{\boldsymbol{q}}_{\boldsymbol{n}} = \begin{bmatrix} \ddot{u}_1 \\ 0 \\ \ddot{u}_3 \end{bmatrix}$, podemos obtener la aceleración medida por el acelerómetro esperada si sumamos la gravedad según la componente 3 negativa y proyectamos sobre {f}:

$$\begin{bmatrix} G_1 \\ 0 \\ G_3 \end{bmatrix}_f = R_{n,f}^{T} * \begin{bmatrix} \ddot{u}_1 \\ 0 \\ \ddot{u}_3 - g \end{bmatrix}_n \qquad \textit{Ecuación 5.8}$$

Resultando las expresiones para aceleración medida:

$$\begin{cases} G_1 = \cos(q_5) * \ddot{u}_1 - \sin(q_5) * (\ddot{u}_3 - g) \\ G_3 = \sin(q_5) * \ddot{u}_1 + \cos(q_5) * (\ddot{u}_3 - g) \end{cases} \qquad \textit{Ecuación 5.9}$$

Estas dos expresiones son las que implementaremos directamente para obtener la medida ideal del acelerómetro a partir de los valores "reales" de $\ddot{u}_1$ y $\ddot{u}_3$.

Si le damos la vuelta a la Ecuación 5.8, podemos obtener la expresión para calcular la aceleración medida por el acelerómetro a partir de los valores G_1 y G_3 ya con el ruido añadido.

$$\begin{cases} \ddot{u}_{1\_acc} = \cos(q_5) * G_1 + \sin(q_5) * G_3 \\ \ddot{u}_{3\_acc} = (-\sin(q_5) * G_1 + \cos(q_5) * G_3) + g \end{cases} \qquad \textit{Ecuación 5.10}$$

La implementación del acelerómetro en simulink la hacemos en tres fases.

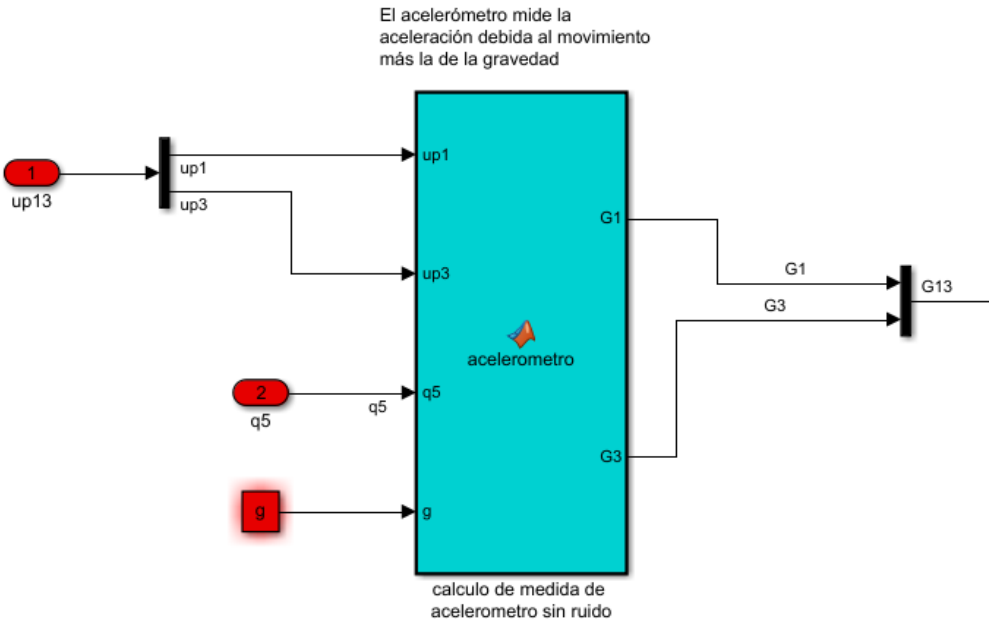


Figura 5.13 Acelerómetro 1, entradas y bloque en el que se calculan las aceleraciones vistas por el sensor

Fuente: elaboración propia

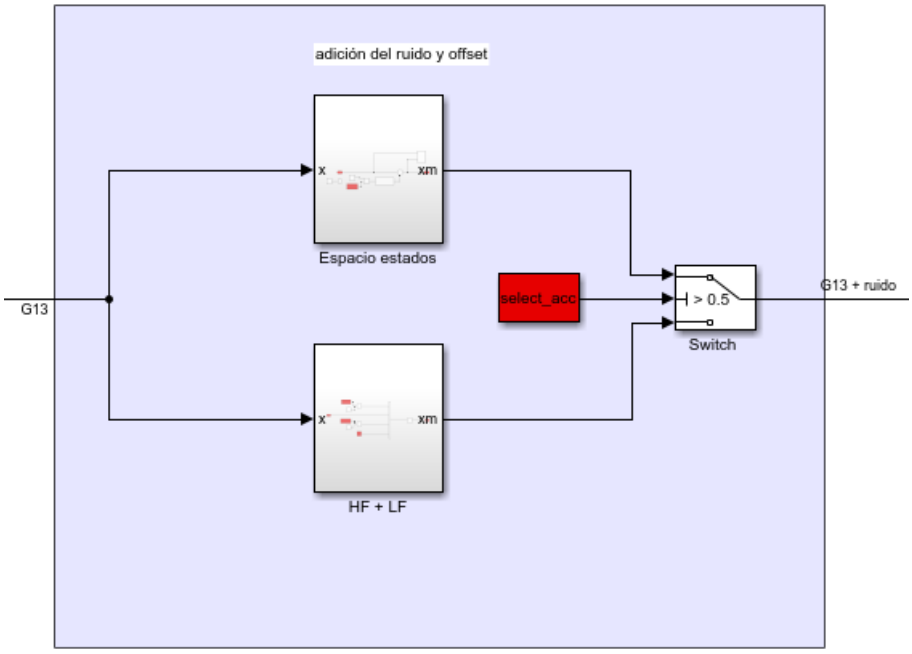


Figura 5.14 Acelerómetro 2, adición del ruido a la medida ideal

Fuente: elaboración propia

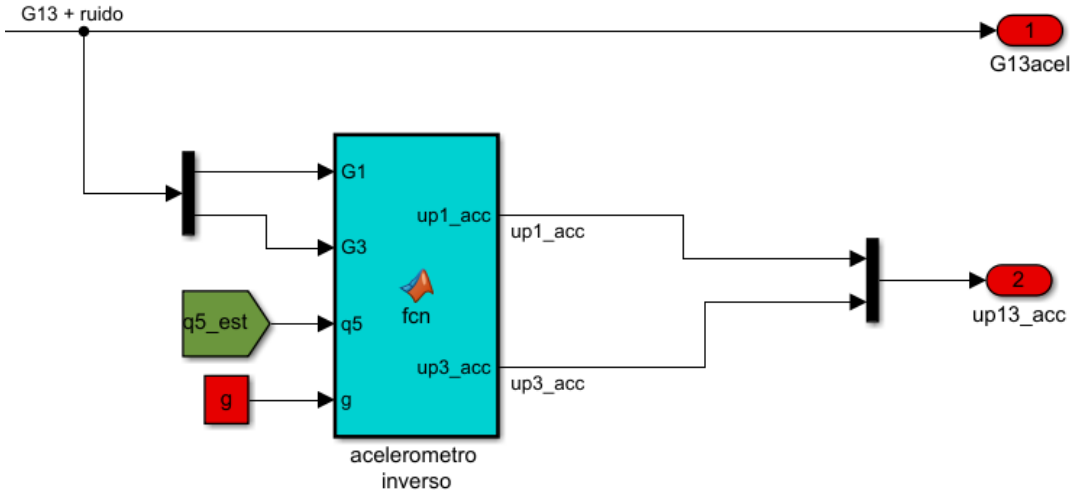


Figura 5.15 Acelerómetro 3, salida de las aceleraciones vistas por el sensor y el cálculo de las aceleraciones globales a partir de estas

Fuente: elaboración propia

En la primera fase de entrada, se computa la Ecuación 5.9 para obtener la medida ideal del sensor. A continuación, a esta señal se le añade en la fase 2 el ruido según los modelos de ruido que se han desarrollado anteriormente. Por último, en la fase 3 se crean dos salidas; la primera es directamente la medida del acelerómetro con ruido, que será utilizada posteriormente en la estimación de la variable q_5. La segunda salida es la medida de la aceleración del robot, obtenida mediante la implementación de las funciones de la Ecuación 5.10.

En una simulación, las medidas del acelerómetro comparadas a las aceleraciones reales tienen la siguiente forma:

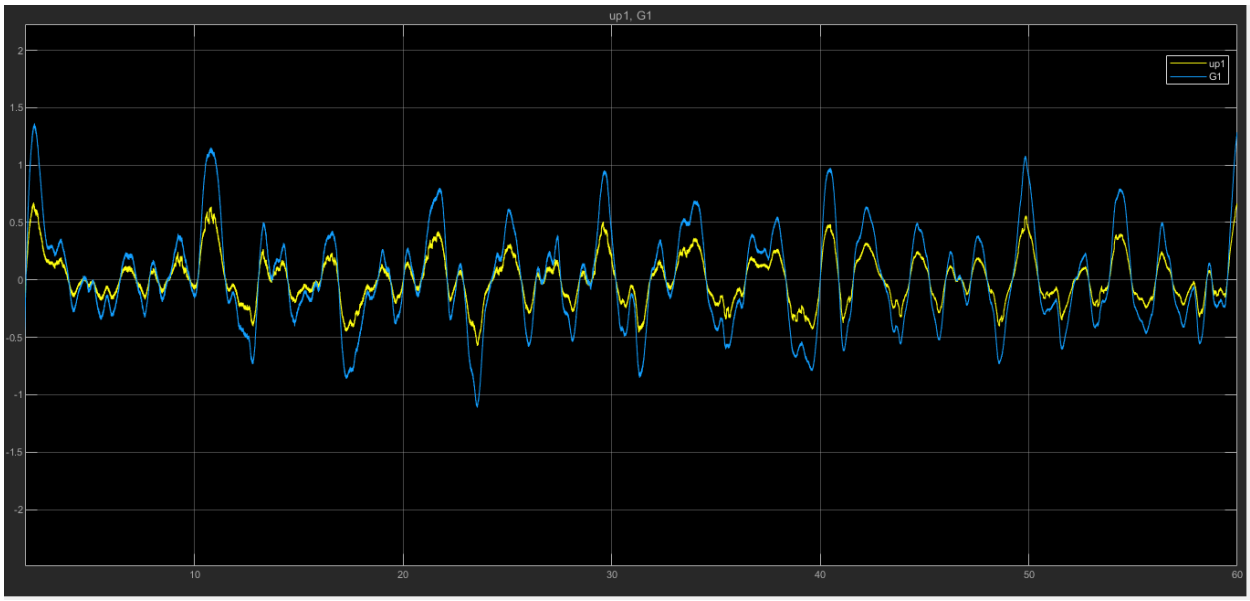


Figura 5.16 G1

Fuente: elaboración propia

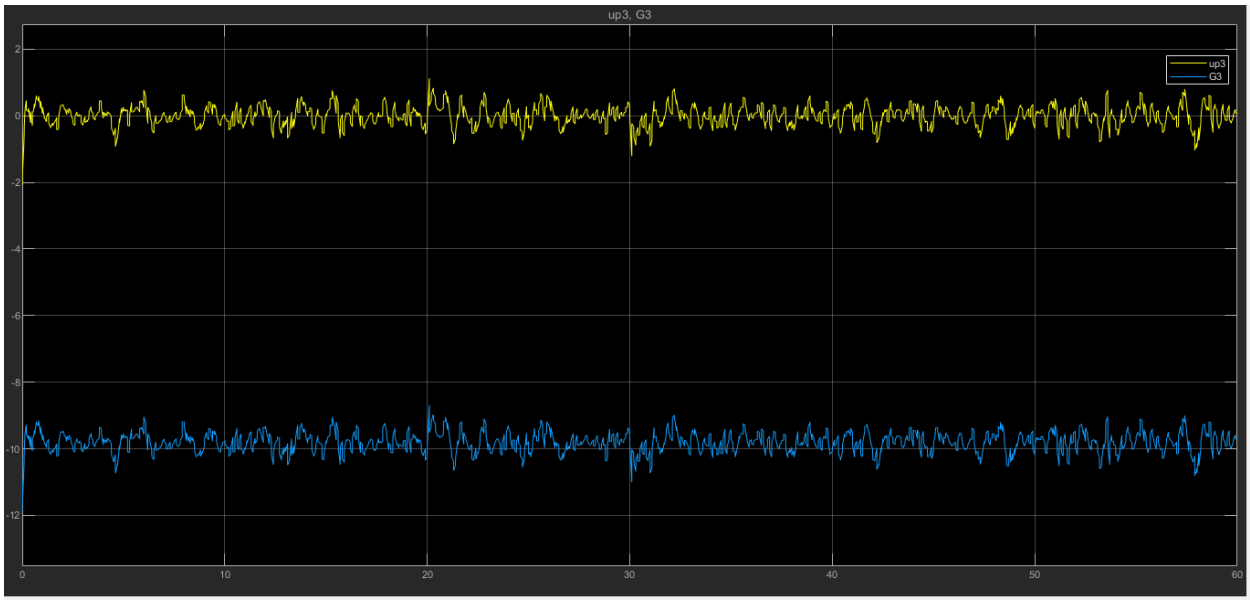


Figura 5.17 G3

Fuente: elaboración propia

En ellas podemos observar que son evoluciones temporales bastante parecidas en el caso de G_1, que discrepa poco con respecto a la original debido a que la proyección de la gravedad es relativamente pequeña en la dirección de f1. EN el caso de G_3, se tiene también una evolución bastante parecida pero desplazada un offset de aproximadamente -9.8, por el efecto de la gravedad que se ve reflejado en el acelerómetro.

5.2.3.1.2 Giróscopo

El giróscopo nos proporciona una medida de velocidad angular en cada uno de sus tres ejes. Es decir, que si se coloca en el centro de gravedad del robot y con los ejes coincidentes con los del sistema de referencia móvil {f}, estaremos recibiendo directamente una medida de u_4, u_5 y u_6. Descartando los giros no permitidos en el plano vertical, la medida que nos interesa de este sensor es la velocidad angular u_5. Para su modelado y simulación, crearemos la señal medida de u_5 a partir de la u_5 "real", que como ocurría en el caso de las aceleraciones es la que se calcula a partir del modelo dinámico del sistema.

Por lo tanto, la implementación de este sensor en Simulink consiste únicamente en sumar el ruido a la variable u_5 "real".

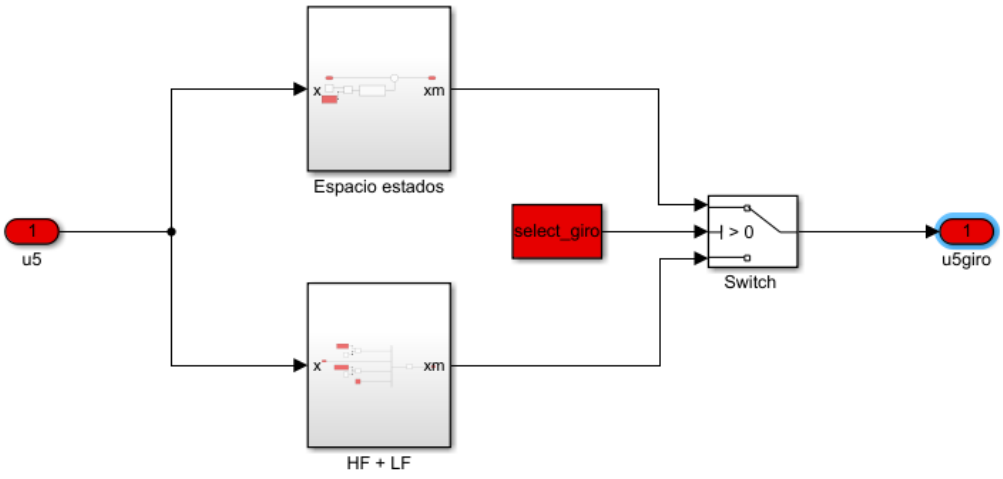


Figura 5.18 Giróscopo

Fuente: elaboración propia

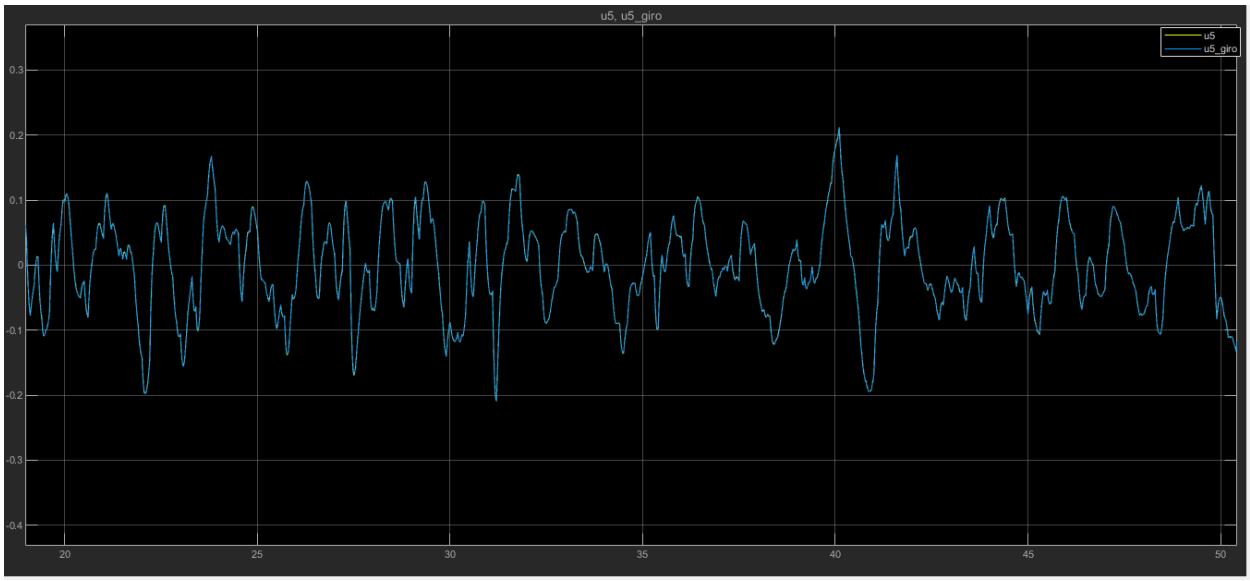


Figura 5.19 Medida del giróscopo

Fuente: elaboración propia

5.2.3.2 GPS

Hasta ahora hemos estado hablando únicamente de GPS para referirnos a este sistema de posicionamiento, pero en realidad en este momento conviene aclarar que nuestro multirrotor irá equipado con un sistema de GPS que proporciona medidas tanto de posición como de velocidad, cada una de ellas con su descripción estadística independiente. Por esta razón vamos a dividir el sensor en dos modelos: GPSp de posición y GPSv de velocidad.

5.2.3.2.1 GPSp

Generalmente, los sensores GPS proporcionan una medida de coordenadas globales que nos obligan a trabajar sobre ciertos sistemas de referencia llamados ECEF[7] y geodésico, que permiten establecer una relación entre coordenadas globales y puntos de la superficie terrestre. Tal y como se estudia en [3], estas medidas pueden ser convertidas a un sistema de referencia local cartesiano mediante expresiones matemáticas conocidas, pero que no entran dentro de los objetivos de este trabajo.

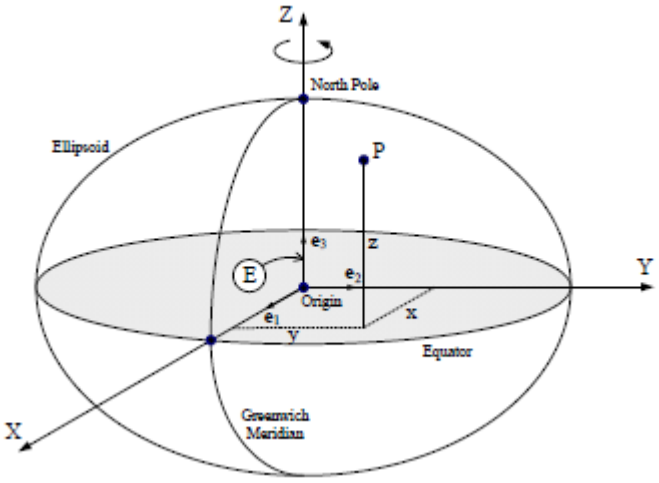


Figura 5.20 Sistema de referencia ECEF

Fuente: [3]

[7] Earth centered, earth fixed

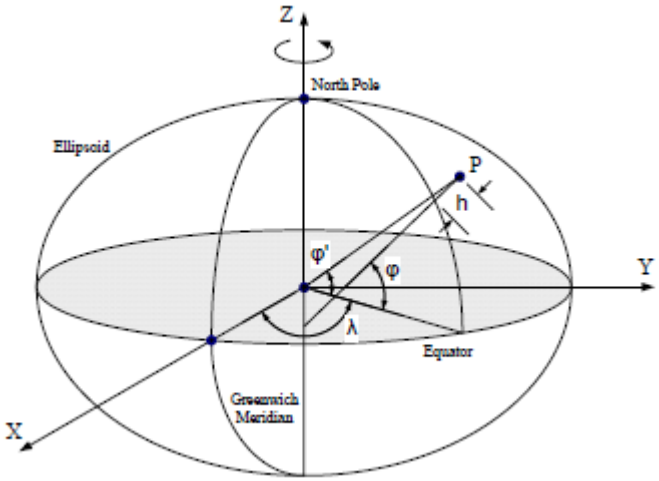


Figura 5.21 Sistema de referencia geodésico

Fuente: [3]

Por tanto, en nuestro caso, implementaremos la señal de medida del GPSp como una suma de la señal "real", obtenida del modelo dinámico del sistema más el ruido generado por nuestros modelos.

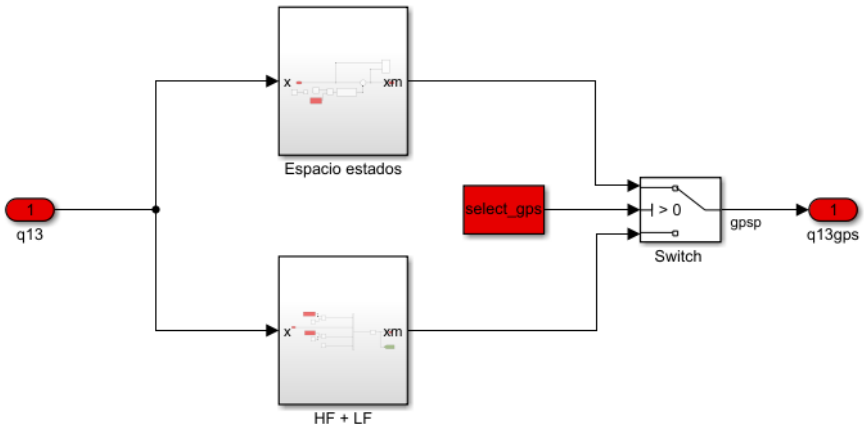


Figura 5.22 GPSp

Fuente: elaboración propia

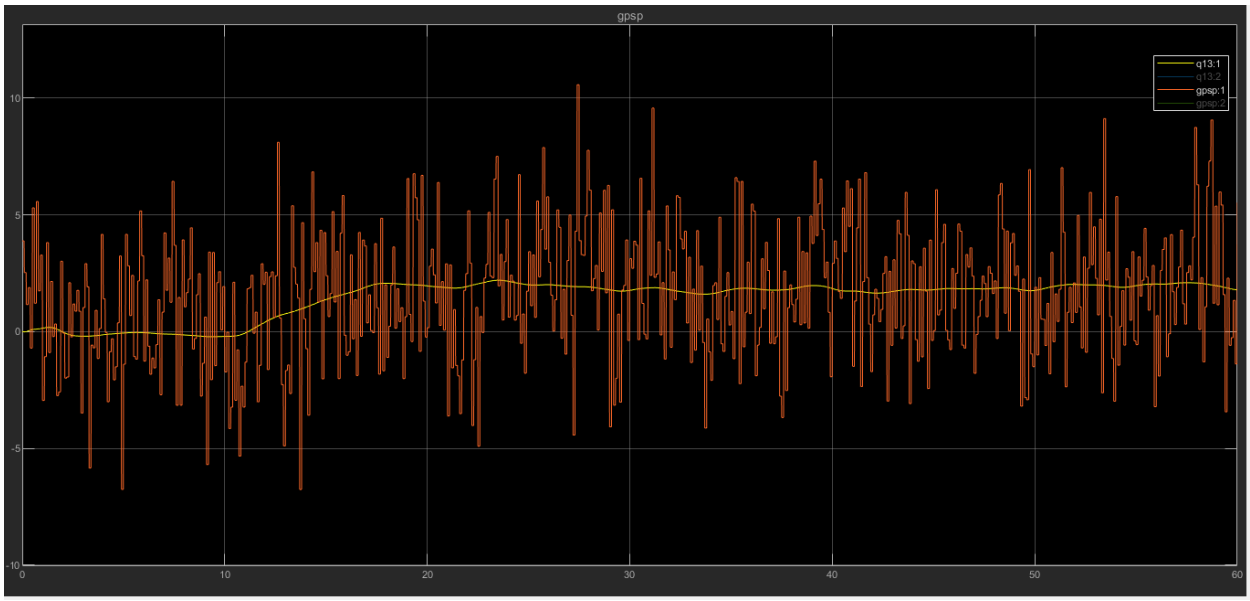


Figura 5.23 Medida del GPSp

Fuente: elaboración propia

5.2.3.2.2 GPSv

Con el GPS en velocidad ocurre algo parecido que con el de posición. Pese a que en la realidad esta medida de velocidad es obtenida por diferenciación a partir de las medidas de posición del propio sistema de posicionamiento, nosotros hemos optado por una implementación en la que simplificamos esta derivación numérica, justificándonos en que es una operación que realiza el propio dispositivo de manera interna. Además, se ha supuesto que las medidas de velocidad y posición son independientes, lo cual de no ser así complicaría la formulación del problema. Por lo tanto, como venimos haciendo hasta el momento, obtenemos la medida sumándole las no idealidades a la velocidad ideal que se obtendría de un GPSv perfecto.

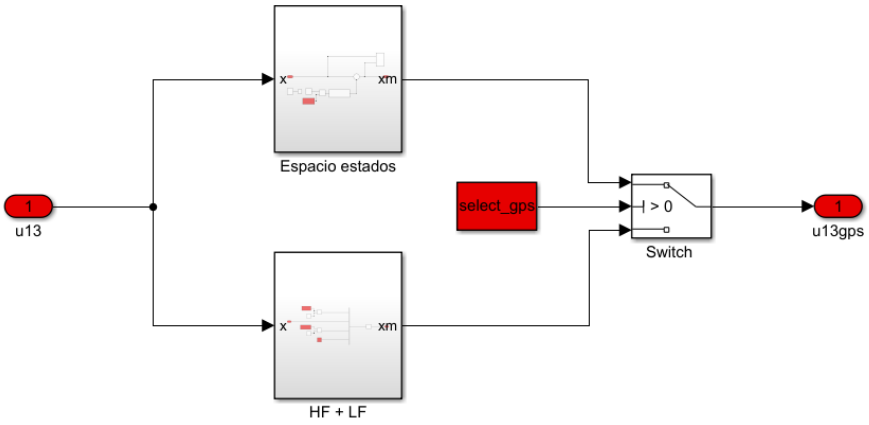


Figura 5.24 GPSv

Fuente: elaboración propia

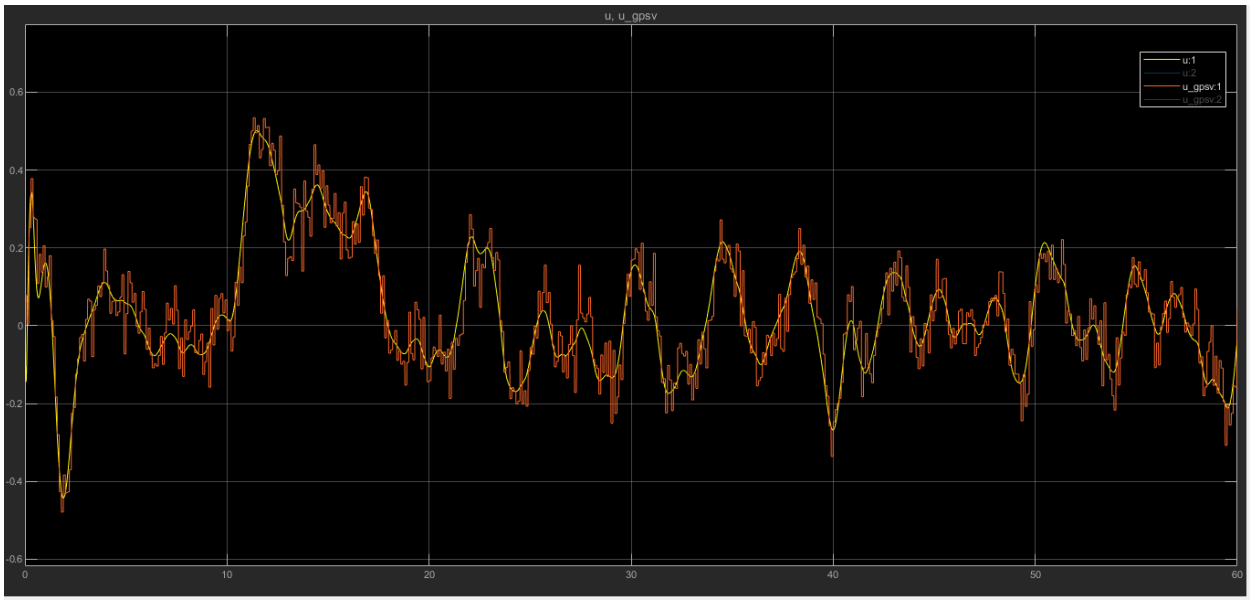


Figura 5.25 Medida del GPSv

Fuente: elaboración propia

5.2.3.3 Altímetro barométrico

El altímetro es un sensor que utilizaremos para obtener una segunda medida de la altura. Este sensor establece una relación entre las condiciones atmosféricas de presión y temperatura y la altura con respecto al nivel del mar. De esta forma, conociendo la localización de nuestro sistema de ejes fijos y la expresión de esa curva característica indicada en la hoja de datos del fabricante podemos calcular la altura a partir del dato de presión medida por el altímetro.

La curva mencionada, siguiendo el planteamiento de [3], es una expresión del tipo:

$$p_m = p_0 \left(1 - A * \frac{q_3}{T_0}\right)^B$$ ***Ecuación 5.11***

Donde p_m es la presión medida por el altímetro, T_0es la temperatura estándar y p_0 es la presión a nivel del mar. Las constantes A y B dependen del dispositivo concreto que estemos utilizando, pero dado que nosotros no vamos a calcular las presiones intermedias, sino que estimaremos la posición directamente como en el resto de los sensores, no es necesario conocerlas. Esto es posible porque a partir del conocimiento de la desviación típica de la medida de presión y de la fórmula que la relaciona con la altura, podemos conocer también la desviación típica del ruido de la medida en altura.

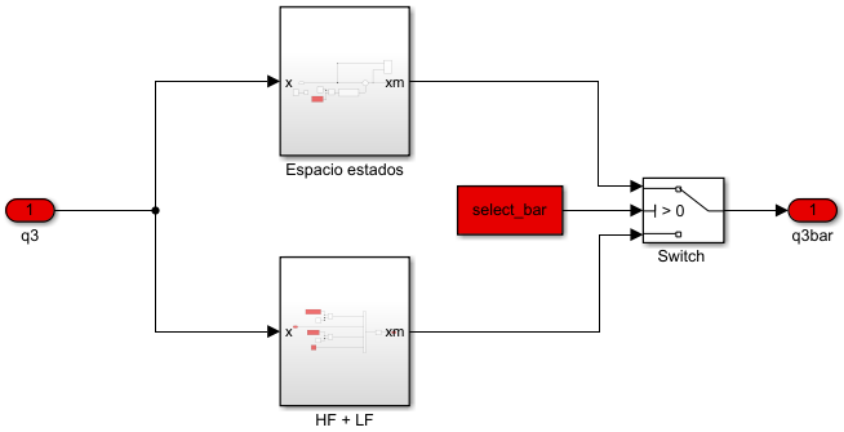


Figura 5.26 Altímetro barométrico

Fuente: elaboración propia

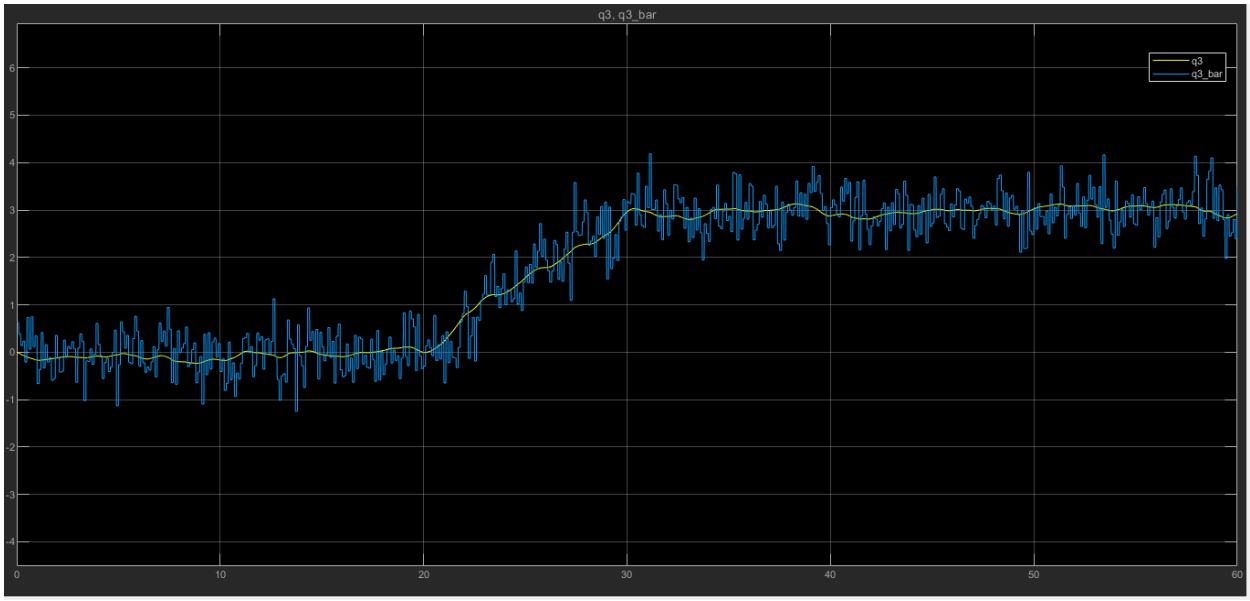


Figura 5.27 Medida del altímetro barométrico

Fuente: elaboración propia

5.2.3.4 Codificadores ópticos

Estos sensores, explicados brevemente en el apartado "5.1 Selección de sensores", proporcionan una medida de la posición articular de los motores del brazo manipulador. Puesto que tienen una frecuencia de muestreo muy alta respecto al resto de sensores utilizados, podemos suponer que proporcionan una medida continua en el tiempo, o en este caso que se actualizan con el mismo periodo que el paso de simulación.

En la realidad, un codificador óptico no puede medir cualquier angulo en los números reales, sino que la circunferencia del disco solidario al eje está muestreada a una determinada resolución. El dispositivo elegido, por ejemplo, tiene una resolución de 1024 bits, lo que significa que nuestro sensor puede discriminar valores de hasta $\frac{360}{1024} \approx 0.352\ grados$. Para simular este comportamiento discreto, se implementa un bloque "Matlab Fcn" con el siguiente código:

```
function q_enc = fcn(q)

res=360/1024;    %grados
res=res*pi/180; %rad

aux=round(q/res);

q_enc=aux*res;
```

Donde la entrada q es un vector con los valores "reales" de q_7 y q_8, y la salida q_{enc} se convertirá directamente en la observación del estado de posición articular tras sumarle el ruido como al resto de sensores.

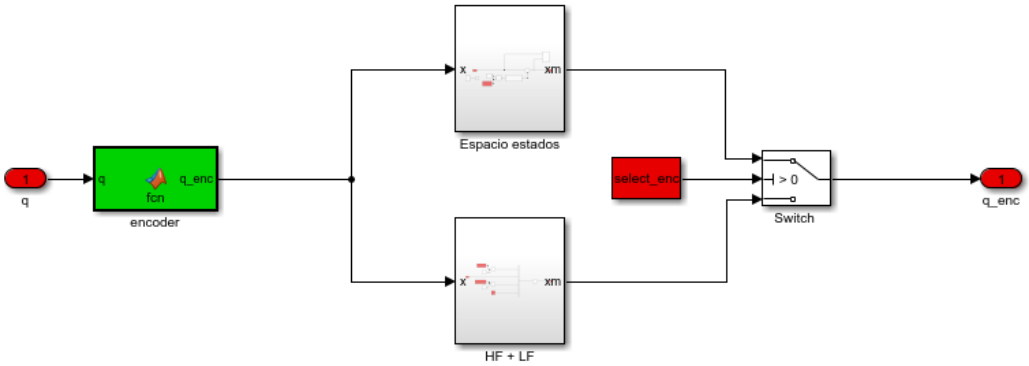


Figura 5.28 Codificador óptico

Fuente: elaboración propia

Necesitamos también una estimación de las velocidades angulares u_7 y u_8 para la realimentación del bucle de control del manipulador. Para ello, al contrario que en el caso del GPS, sí debemos abordar la derivación numérica puesto que esto no es algo que se haga a nivel de dispositivo, sino que suele ser implementado a nivel de software.

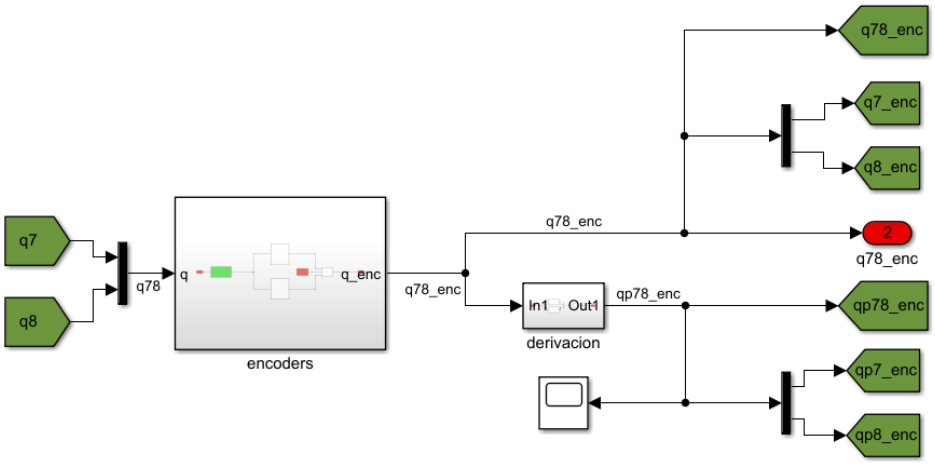


Figura 5.29 Velocidades angulares u7 y u8

Fuente: elaboración propia

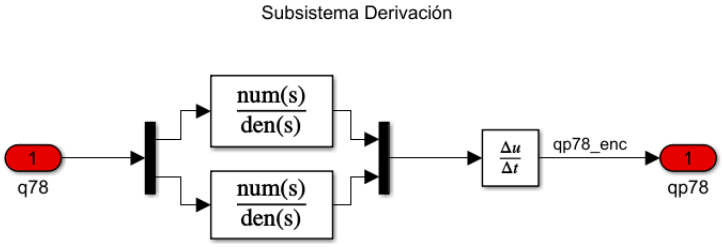


Figura 5.30 Derivación de q7 y q8

Fuente: elaboración propia

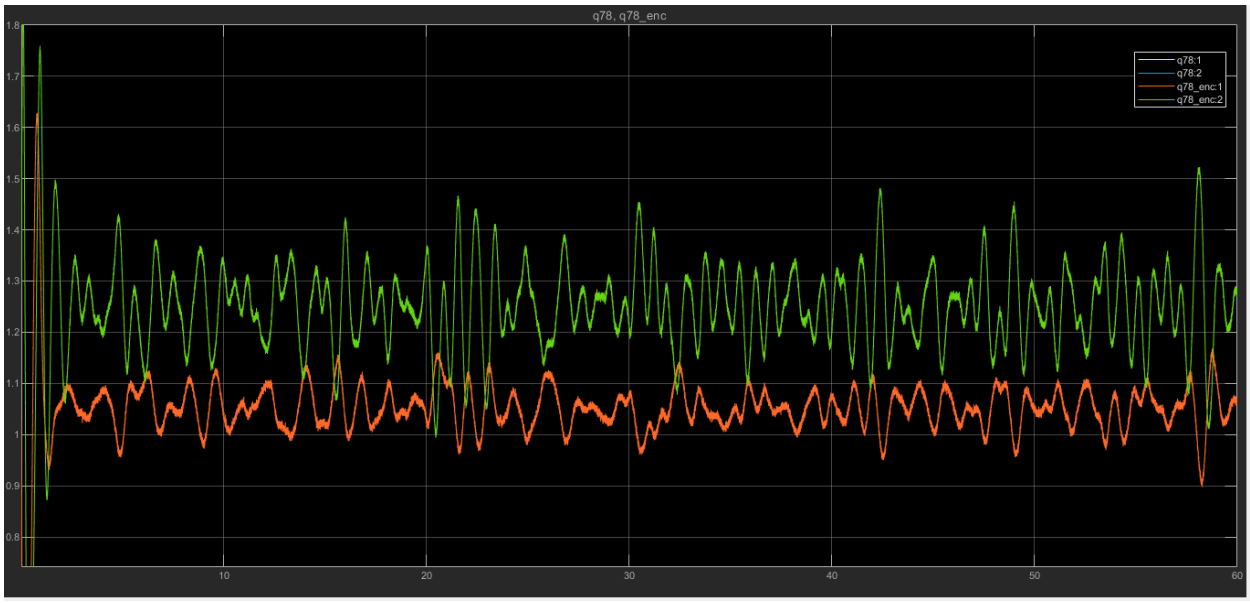


Figura 5.31 Desempeño de los codificadores ópticos

Fuente: elaboración propia

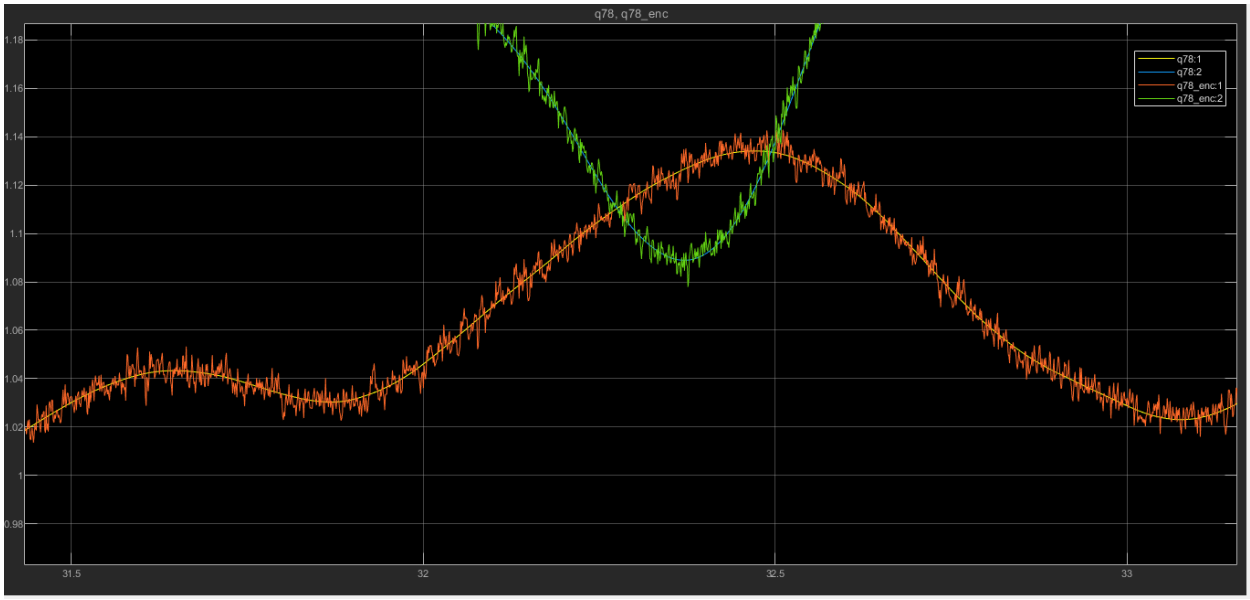


Figura 5.32 Zoom de la medida de los codificadores

Fuente: elaboración propia

En las gráficas anteriores se puede observar cómo las medidas obtenidas a través de los codificadores ópticos se encuentran situadas justo sobre la posición articular "real", con su correspondiente ruido añadido.

5.2.3.5 Parámetros de los sensores

Se indican en una tabla, a modo de resumen todos los valores numéricos de los parámetros de los sensores elegidos.

Sensor		Modelo real	F (Hz)	σ		Fuente
				Typ.	Uds.	
IMU	Acelerómetro	ICM-20602	100	9.8e-3	m/s^2	[14]
	Giróscopo		100	0.04*π/180	rad/s	
GPS	GPSp	u-blox NEO M8	10	2.5	m	[15]
	GPSv		10	0.05	m/s	
Altímetro		BOSCH BMP 180	10	0.4	m	[3],[17]
Codificadores ópticos		Faulhaber IER3	250e3	0.3*pi/180	rad	[16]

Tabla 5-1 Parámetros de los sensores

6 ESTIMACIÓN DE ESTADO, FILTRO DE KALMAN

En esta sección trataremos de obtener una estimación del estado de cada una de las variables cinemáticas de nuestro sistema. Para el giro y velocidad de la plataforma con respecto al sistema de referencia fijo, se ha estudiado el comportamiento con un filtro complementario y con un filtro de Kalman, y como se verá en una comparativa, se opta por la implementación con el filtro de Kalman. También será un KF para el resto de las variables de posición y velocidad del UAV. Por último, no necesitaremos estimar el estado de los motores del manipulador, puesto que tomaremos directamente la medida de los codificadores ópticos.

6.1 Estimación de posición y velocidad

En esta sección vamos a implementar el algoritmo del filtro de Kalman (KF) recursivo para la estimación del estado de posición y velocidad del UAV, basándonos en los contenidos de [13]. El KF, desarrollado por Rudolf E. Kalman, es un algoritmo de estimación de estado muy ampliamente utilizado en muchas ramas de la ingeniería y especialmente en control. Dicho algoritmo tiene aplicación y es **óptimo** en sistemas dinámicos lineales, para los que se pretende estimar el estado a priori desconocido a partir de medidas que pueden contener ruido gaussiano y otras desviaciones. La descripción estadística de este ruido es contemplada en la implementación del filtro mediante la matriz de covarianza de la medida.

El estado que pretendemos observar es el siguiente:

$$x_k = \begin{bmatrix} q_1 \\ q_3 \\ u_1 \\ u_3 \\ \dot{u}_1 \\ \dot{u}_3 \end{bmatrix} = A * x_{k-1} + B * u_k + \epsilon_k$$

Ecuación 6.1

Donde ϵ_k, vector de tamaño 6x1 como x_k es el ruido de la estimación e irá implementado en la matriz de confianza en el modelo R_k, que definiremos como:

$$R_k = trust^2 * I_{6x6}$$

Ecuación 6.2

Siendo R de tamaño 6x6 y "trust" un parámetro a elegir, inversamente proporcional a la confianza que tengamos en la predicción.

La aplicación del filtro de Kalman se realiza en dos etapas. En la primera de ellas, llamada etapa de predicción, tratamos de estimar un estado del sistema que llamaremos $x_{pred,k}$. Esta estimación a priori, hecha en cada paso de simulación, la realizaremos mediante ecuaciones básicas de la cinemática, sin atender a lo que ocurre en la realidad. Las ecuaciones del filtro de Kalman que rigen la predicción son las siguientes:

$$x_{pred,k} = A * x_{k-1} + B * u$$

Ecuación 6.4

Donde x_{k-1} es el estado estimado en el ciclo anterior, u es el vector de actuación, y A y B son dos matrices que vamos a identificar a continuación.

Nuestra predicción la vamos a realizar mediante las ecuaciones más básicas de la cinemática suponiendo que la aceleración entre un ciclo y el siguiente es constante (MRUA[8]):

[8] Movimiento Rectilíneo Uniformemente Acelerado

$$\begin{cases} x_k = x_{k-1} + \dot{x_{k-1}} * \Delta T + \frac{1}{2} * \ddot{x_{k-1}} * \Delta T^2 \\ \dot{x_k} = \dot{x_{k-1}} + \ddot{x_{k-1}} * \Delta T \\ \ddot{x_k} = \ddot{x_{k-1}} \end{cases}$$ ***Ecuación 6.3***

Podemos identificar términos, obteniendo las matrices A y B:

$$A = \begin{bmatrix} 1 & 0 & \Delta T & 0 & 0.5 * \Delta T^2 & 0 \\ 0 & 1 & 0 & \Delta T & 0 & 0.5 * \Delta T^2 \\ 0 & 0 & 1 & 0 & \Delta T & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$ ***Ecuación 6.6***

$$B = 0$$ ***Ecuación 6.7***

No utilizamos la señal de actuación *u* en la predicción.

A continuación, la matriz de covarianza del proceso a priori:

$$P_{pred,k} = A * P_{est,k-1} * A^T + R_k$$ ***Ecuación 6.8***

Más tarde, se pasa a la segunda etapa, llamada de corrección o de actualización. En nuestra implementación, esta etapa se ejecutará a la frecuencia a la que nuestros sensores proporcionen una medida. Un factor a tener en cuenta es que la frecuencia a la que la IMU y el GPS proporcionan medidas no es la misma, por lo que tenemos que adaptar la ejecución a alguna de ellas. Existen varias posibilidades:

- Tomar la frecuencia más rápida de los sensores. En este caso ejecutaríamos la fase de actualización cada vez que tuviéramos una medida de la IMU, que funciona a 100 Hz frente a los 10 Hz del GPS y el altímetro. Para las correcciones de los sensores más lentos, supondríamos que la medida proporcionada por estos en todos los ciclos intermedios es constante.
- Tomar la frecuencia más baja de los sensores. De esta forma estaríamos ejecutando la fase de corrección siempre que tuviéramos una medida de los sensores lentos (a 10 Hz), mientras que todas las medidas intermedias que proporciona la IMU estarían siendo desechadas.
- Ajustar el filtro y realizar correcciones distintas en función de la frecuencia de cada medida. Este último método aprovecharía todas las medidas de nuestros sensores, pero haría el algoritmo más complicado, no estando realmente demostrada una gran mejora frente a tomar alguna de las dos opciones anteriores.

Optaremos por la segunda opción, ejecutando la fase de corrección a la frecuencia del sensor más lento. En la fase de actualización, se computa la llamada "ganancia de Kalman" (matriz K) que modifica la estimación de cada variable hacia un nuevo valor, de forma ponderada según la confianza que tengamos en el proceso de estimación o en la medida de los sensores. Esta corrección, además, es proporcional a la discrepancia que existe entre el estado estimado y el estado medido por los sensores, conocida también por el nombre de "innovación".

El vector de medidas que utilizaremos para computar la innovación es el siguiente:

$$z_k = C * x_k + \delta_k$$ ***Ecuación 6.9***

El vector z, puesto que para la estimación de la altura tenemos dos medidas en lugar de una, tendrá una componente más que x, siendo un vector 7x1:

$$z_k = \begin{bmatrix} q_{1,gpsp} \\ q_{3,gpsp} \\ u_{1,gpsv} \\ u_{3,gpsv} \\ q_{3,bar} \\ \dot{u}_{1,acc} \\ \dot{u}_{3,acc} \end{bmatrix}$$ ***Ecuación 6.10***

Podemos identificar la matriz C.

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$ ***Ecuación 6.11***

Y el vector δ_k, que representa al ruido de la medida quedaría implementado en la matriz de covarianza de la medida, Q:

$$Q = \begin{bmatrix} \sigma_{gpsp}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{gpsp}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{gpsv}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{gpsv}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{bar}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{acc}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{acc}^2 \end{bmatrix}$$ ***Ecuación 6.12***

Con lo que ya podemos pasar a la fase de corrección. La ganancia de Kalman:

$$K_k = P_{pred,k} * C^T * \left(C * P_{pred,k} * C^T + Q\right)^{-1}$$ ***Ecuación 6.13***

El estado estimado, se corrige con la ganancia de Kalman y con la innovación:

$$x_{est,k} = x_{pred,k} + K * (z_k - C * x_{pred,k})$$ ***Ecuación 6.14***

Y por último se actualiza la matriz de covarianza para el siguiente ciclo:

$$P_{est} = (I - K * C) * P_{pred}$$ ***Ecuación 6.15***

En todos los ciclos en que se ejecute el programa del filtro (una vez cada paso de simulación) se pasará por la fase de predicción, mientras que la fase de actualización solo la ejecutaremos cuando tengamos nueva medida en GPS y altímetro, desechando todas las medidas de la IMU intermedias (1 útil de cada 10).

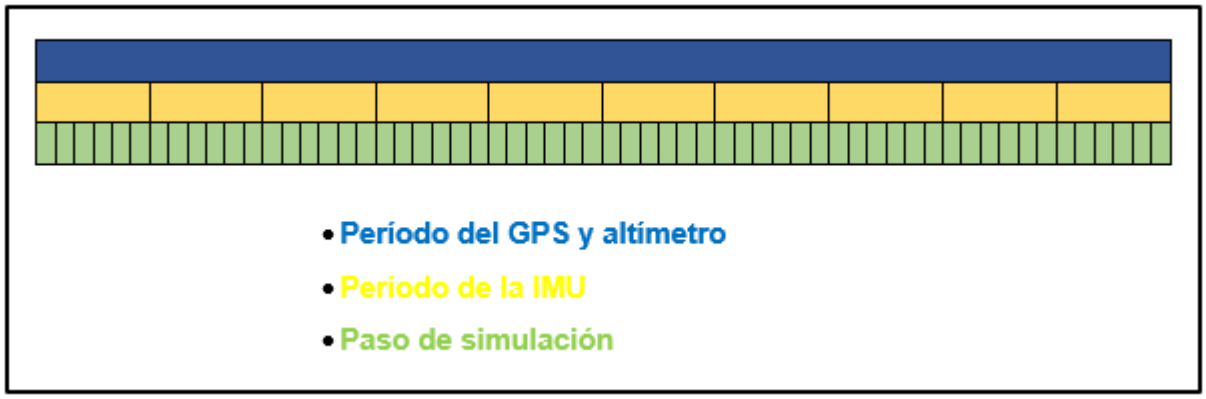


Figura 6.1 Esquema de períodos de sensores

Fuente: elaboración propia

Para ello, necesitamos generar una variable "flag_med" que determine cuándo recibimos una nueva medida. Esto lo haremos en simulink de la siguiente forma:

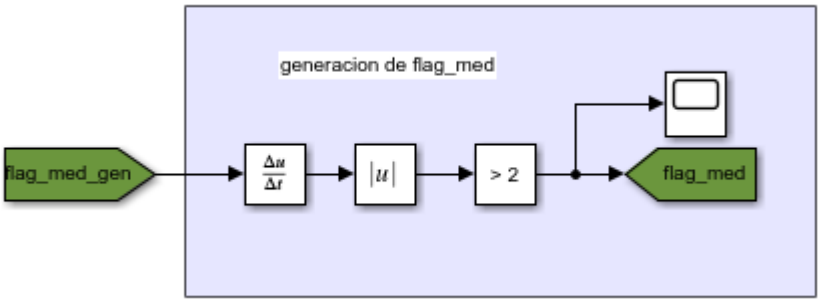


Figura 6.2 Generación de flag_med

Fuente: elaboración propia

Donde la entrada "flag_med_gen" proviene de justo después del mantenedor de orden cero del GPS que podemos observar en la Figura 5.3 Modelo del ruido HF + LF. Aunque estemos simulando con el modelo de filtro de primer orden de los sensores no habrá problemas en la generación de la variable flag_med, puesto que ambos modelos de sensores se computan independientemente de cuál de los dos se esté utilizando.

El KF será implementado en Simulink con un bloque Matlab Fcn, con las siguientes entradas y salidas:

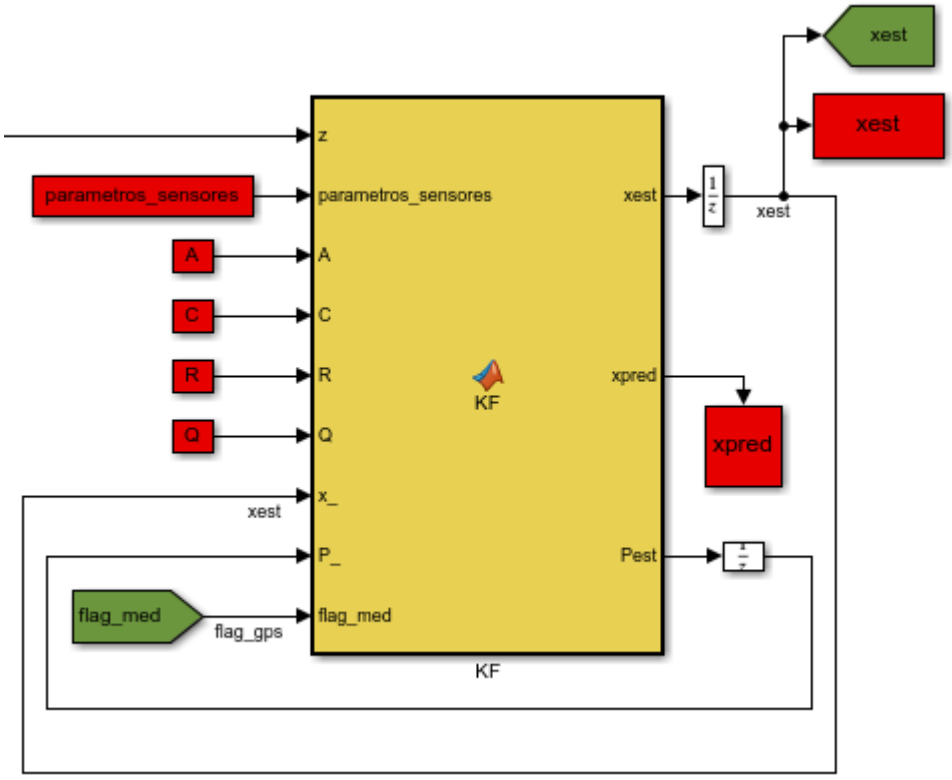


Figura 6.3 Bloque del Filtro de Kalman

Fuente: elaboración propia

Y el código programado en él es el siguiente:

```
function [xest,xpred,Pest]=
KF(z,parametros_sensores,A,C,R,Q,x_,P_,flag_med)
%Implementación de un EKF para estimación de estado
%% Inicialización
xpred=zeros(6,1);
xest=zeros(6,1);
Ppred=zeros(6,6);
Pest=zeros(6,6);

%% FASE DE PREDICCIÓN (a la frecuencia del paso de simulación)
xpred=A*x_;
Ppred=A*P_*A' + R;

%% Fase de corrección (a la freciencia de medidas de GPS y BAR)
if flag_med==1

    K=Ppred*C' / (C*Ppred*C' + Q);
    xest=xpred+K*(z-C*xpred);
    Pest=(eye(6)-K*C)*Ppred;

else

    xest=xpred;
    Pest=Ppred;

end
```

Con esto, podríamos realizar ya simulaciones para comprobar el funcionamiento del KF, únicamente a falta de

elegir el parámetro "trust". Este valor, como se comentó, es inversamente proporcional a la confianza que tengamos en el modelo de predicción. Es habitual en la práctica elegirlo por un método heurístico, de forma que se elige un valor inicial y según los resultados se modifica. El criterio es, que cuanto menor sea "trust", mejor estamos suponiendo que es la predicción, y por lo tanto el filtro de Kalman hará más caso a la predicción y corregirá menos en la fase de actualización. El valor que se ha encontrado óptimo para este parámetro es:

$$trust = 0.001$$

Realizamos una simulación en la que se han establecido como referencias unas entradas en rampa, de 2 metros para q_1 y de 3 metros para q_3, y todas las demás variables se dejan en su posición inicial. Ya se ha realimentado el sistema directamente con las señales estimadas por el filtro, en lugar de las "reales" que se obtienen del modelo dinámico. El desempeño es el siguiente:

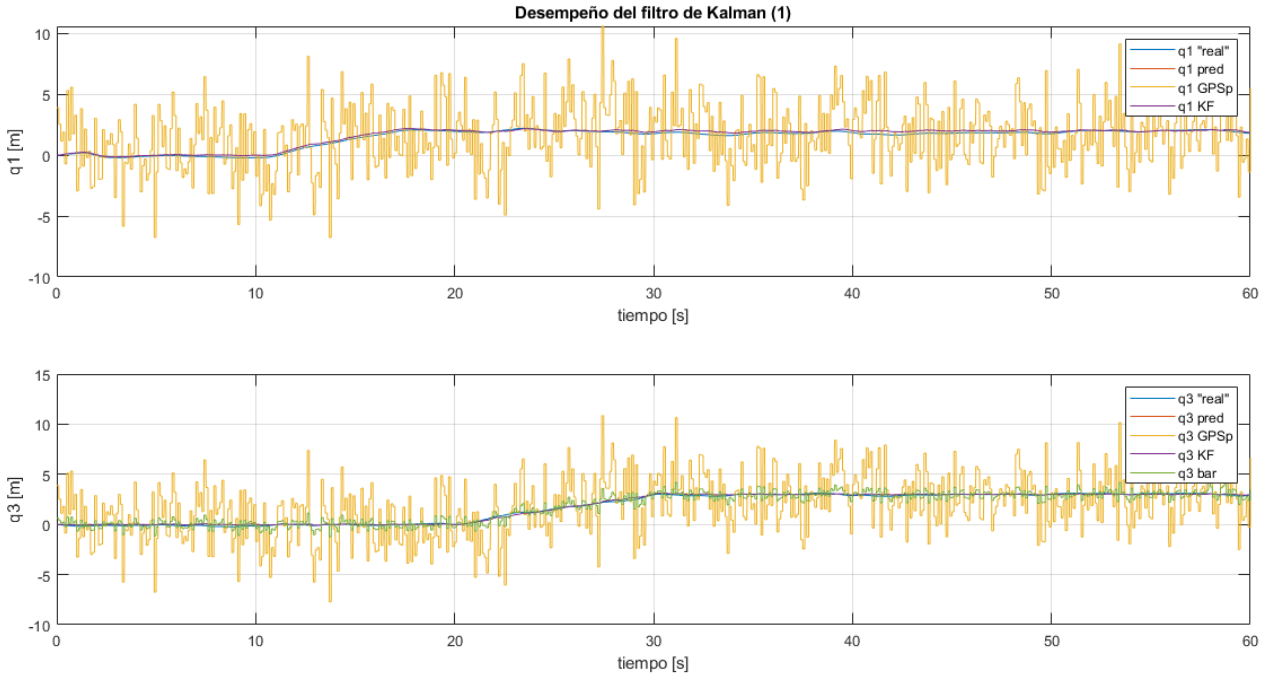


Figura 6.4 Simulación del filtro de Kalman, 1

Fuente: elaboración propia

En las leyendas, "real" denota a las obtenidas por el modelo dinámico, "pred" son aquellas que salen directamente de la fase de predicción del filtro, las que llevan nombres de sensores son las respectivas medidas y por último "KF" son las estimadas por el filtro.

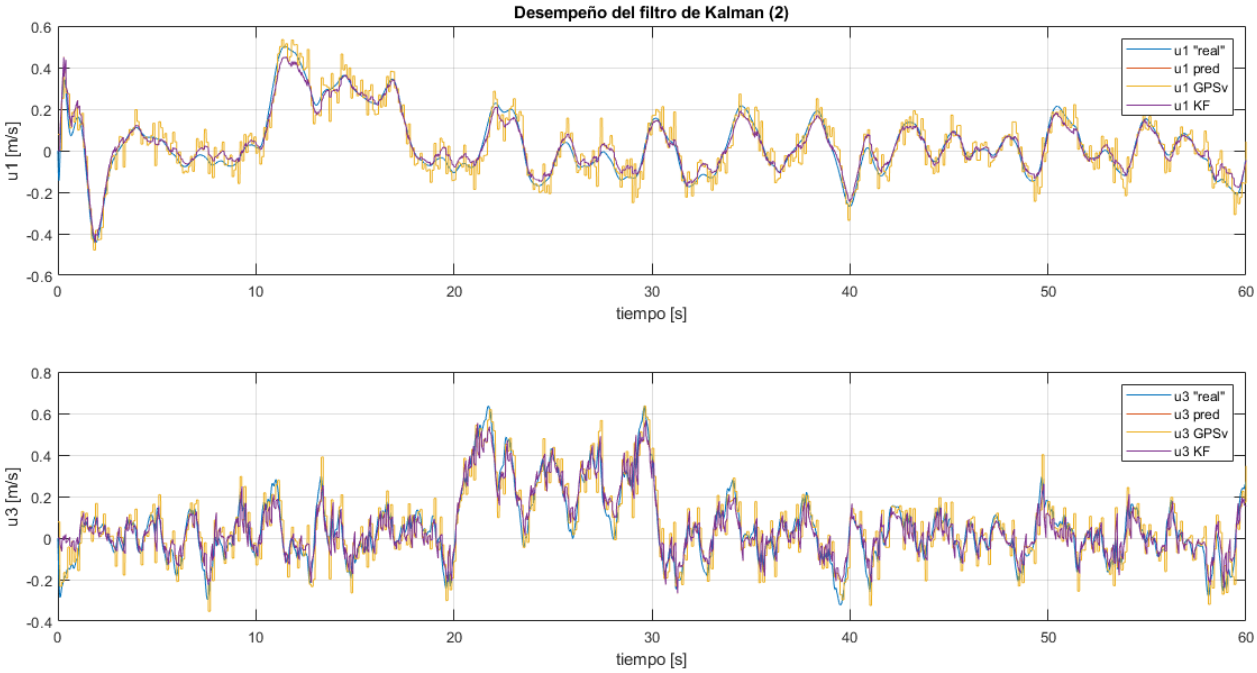


Figura 6.5 Simulación del filtro de Kalman, 2

Fuente: elaboración propia

Observamos como la señal en color morado (estimación del filtro) se ajusta en gran medida a la de color azul (posición "real"), especialmente en el caso de las variables de posición, que varían menos que las de velocidad.

Podemos entender un poco mejor cómo funciona el algoritmo, si cambiamos el parámetro trust a un valor más pequeño, por ejemplo 0.0001, de forma que el filtro tenga menos en cuenta las medidas a la hora de realizar su estimación.

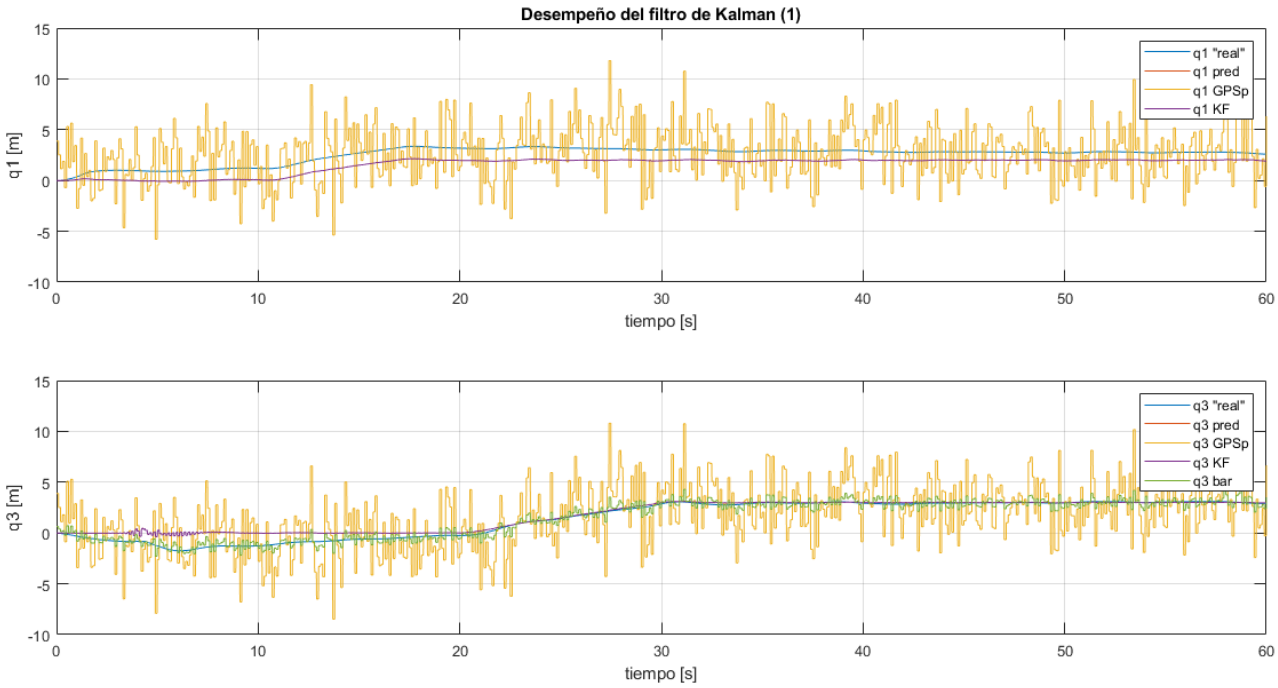


Figura 6.6 Simulación 2 KF, 1

Fuente: elaboración propia

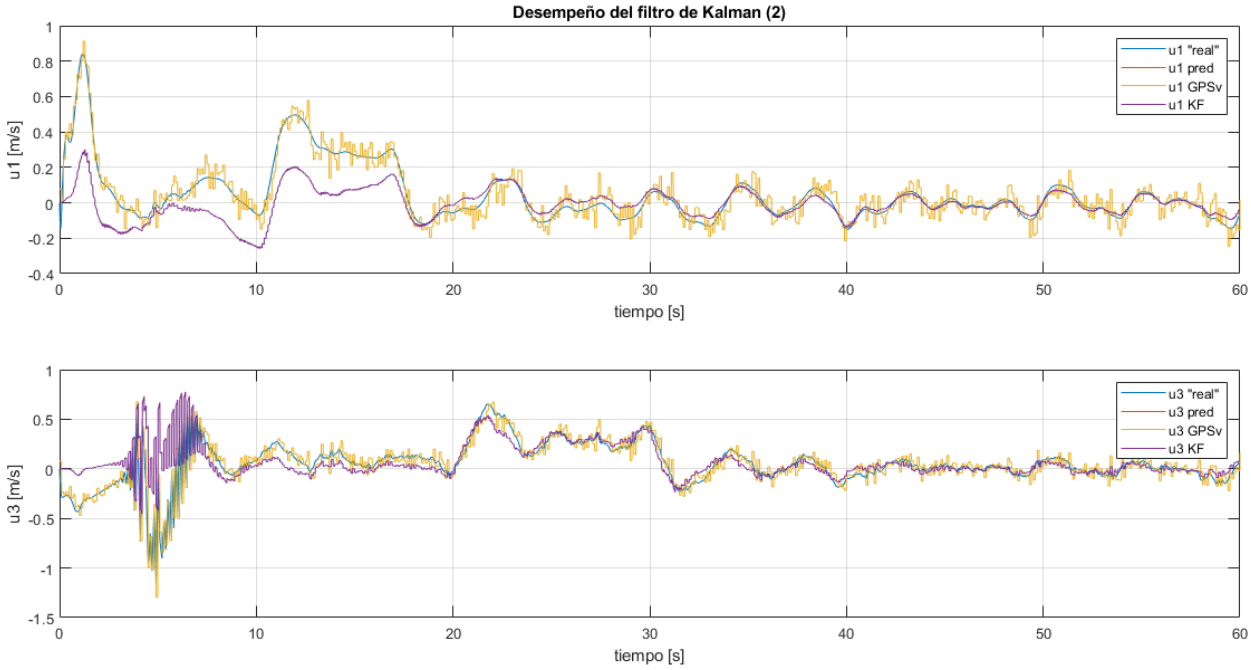


Figura 6.7 Simulación 2 KF, 2

Fuente: elaboración propia

Podemos comprobar que el resultado es peor que en el primer caso que hemos considerado. Si nos fijamos por ejemplo en el comportamiento de q_1, se ve que hay una discrepancia entre la posición real y la que estima el filtro, debido a que para la estimación se está teniendo mucho más en cuenta la predicción, que consiste en integrar las variables cinemáticas (lo cual va acumulando error). En un caso normal ese error acumulado o deriva se corrige en la fase de actualización gracias a las medidas de los sensores, pero al ser el parámetro trust tan pequeño, esta corrección se hace en un grado muy pequeño. En q_3, la corrección de los sensores sí consigue rectificar a la predicción, gracias a la disponibilidad de dos sensores para ello.

Este empeoramiento se manifiesta aún más en la estimación de las velocidades, dado a que la estimación de estas se hace únicamente integrando una aceleración que en el modelo de predicción de la Ecuación 6.3 supusimos constante.

Si aumentáramos el parámetro trust, la estimación del filtro seguiría de forma más exagerada a los sensores, lo cual generaría una señal algo ruidosa y que tiende a desestabilizar el sistema, por lo que concluimos que el valor de $trust = 0.001$ era correcto.

6.2 Estimación de giro de la plataforma

En esta sección se van a realizar las implementaciones del filtro complementario y del filtro de Kalman para compararlas y elegir la que mejor desempeño proporcione en el caso de q_5 y u_5.

El primer paso para cualquiera de las dos es obtener la estimación del ángulo de giro a partir del acelerómetro, $q_{5_{acc}}$. Recordamos que las variables G_1 y G_3 del acelerómetro proporcionan una medida de la aceleración debida al movimiento más la de la gravedad vista en los ejes $\{f\}$. Si suponemos un estado estacionario en el que el UAV no se mueve (aceleración debida al movimiento nula) podemos proyectar el vector de la gravedad sobre los ejes móviles y establecer una relación trigonométrica entre q_5 y las medidas G_1 y G_3.

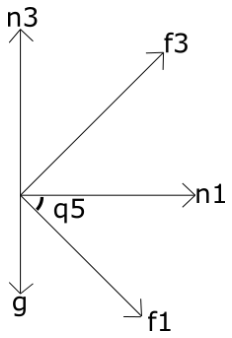


Figura 6.8 Esquema de g, {f} y {n}

Fuente: elaboración propia

$$\frac{G_1}{G_3} = -\tan(q_5) \quad \rightarrow \quad q_{5,acc} = \operatorname{atan}\left(-\frac{G_1}{G_3}\right) \qquad \textit{Ecuación 6.4}$$

La implementación de esta relación en simulink:

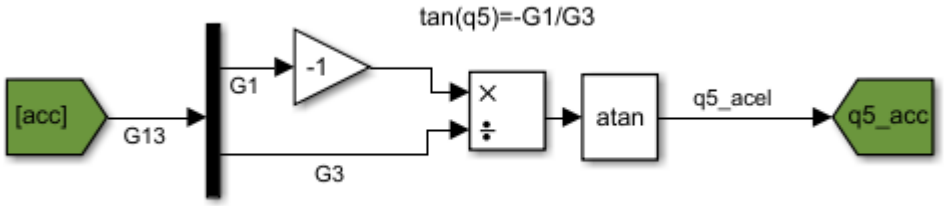


Figura 6.9 Cálculo de q5 con acelerómetro

Fuente: elaboración propia

6.2.1 Filtro complementario

Como ya se mencionó anteriormente, el filtro complementario es un estimador de estado gaussiano derivado del filtro de Kalman, que se usa para fusionar las medidas de q_5 obtenidas al integrar el giróscopo y mediante relaciones trigonométricas de las medidas del acelerómetro.

La expresión que rige la estimación de q_5 es la siguiente:

$$q5_{fc,k} = (1-\alpha) * \left(q5_{k-1} + u_{5,giro,k} * \Delta T\right) + \alpha * q5_{acc,k} \qquad \textit{Ecuación 6.5}$$

Donde $\alpha \sim 0.1, 0.2$ es un parámetro a elegir (0.1 en nuestro desarrollo). Puesto que el filtro complementario va a ser ejecutado en cada paso de simulación, ΔT valdrá ese mismo paso. La frecuencia a la que la IMU proporciona una medida de u_5 o q_5 será naturalmente menor que la inversa del paso de simulación, pero esto no será problema gracias al mantenedor de orden cero o el filtro de primer orden que tenemos a la salida de nuestros sensores, que nos aseguran que existe un valor de medida para todo t.

Luego, para la estimación mediante filtro complementario se computa con un bloque Matlab Fcn la Ecuación 6.5.

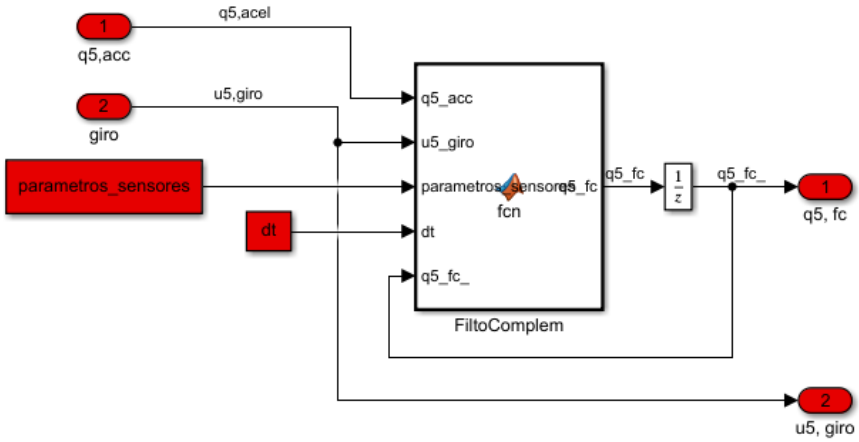


Figura 6.10 Interior del subsistema Filtro Complementario

Fuente: elaboración propia

En una simulación, podemos observar el desempeño del filtro complementario en la estimación de q_5.

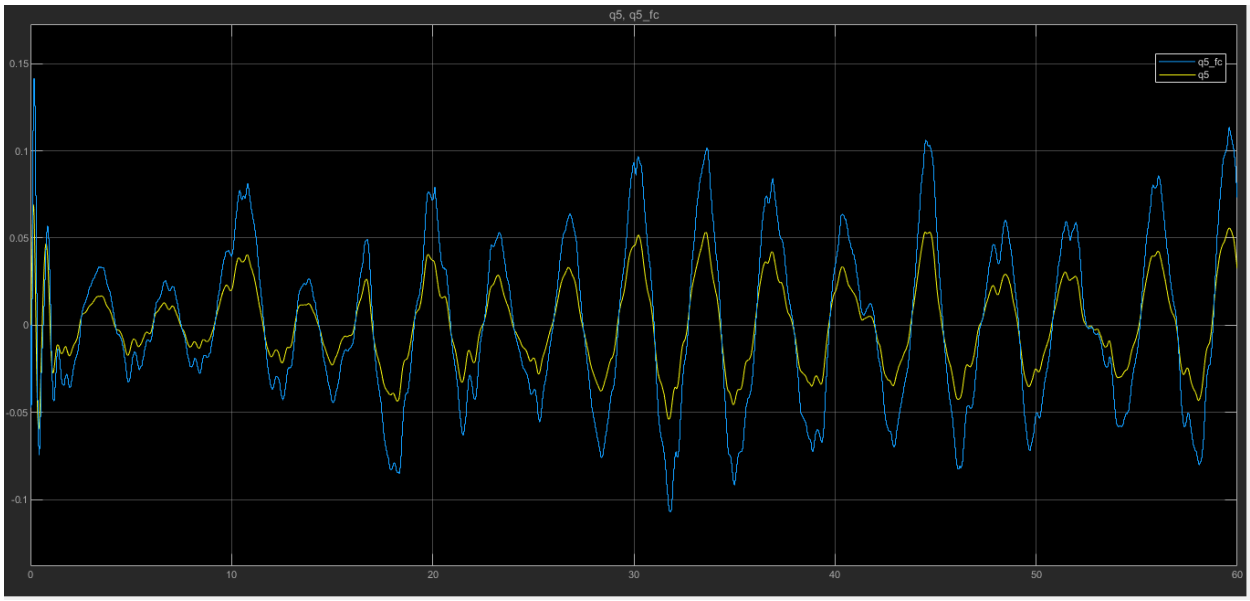


Figura 6.11 Desempeño del filtro complementario para la estimación de q5

Fuente: elaboración propia

6.2.2 Filtro de Kalman

El algoritmo que implementamos en este caso es completamente análogo que el desarrollado en la sección "6.1 Estimación de posición y velocidad". La diferencia más notoria entre ambos es que esta vez la fase de corrección se ejecuta en todos los ciclos, y no a una frecuencia menor que la de predicción.

El estado a estimar:

$$x = \begin{bmatrix} q_5 \\ u_5 \end{bmatrix}$$ *Ecuación 6.6*

La predicción se hará suponiendo que la velocidad de giro es constante entre un paso de simulación y el siguiente y sin tomar en cuenta las señales de actuación, por lo que las matrices A y B quedarían de la siguiente forma:

$$A = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}$$ ***Ecuación 6.7***

$$B = 0$$ ***Ecuación 6.8***

Donde ΔT de nuevo es el paso de simulación.

En la matriz de confianza de la estimación volvemos a utilizar el mismo concepto del parámetro "trust", que de hecho tiene el mismo valor.

$$R = trust * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad trust = 0.001$$ ***Ecuación 6.9***

En la fase de corrección utilizamos el vector de medidas siguiente:

$$z = \begin{bmatrix} q_{5,acc} \\ u_{5,giro} \end{bmatrix}$$ ***Ecuación 6.10***

Puesto que las medidas se corresponden directamente al vector de estado, la matriz de observación C será una identidad de tamaño 2x2.

La matriz Q contiene en su diagonal las varianzas de las medidas. El caso de u_5 es más sencillo, puesto que conocemos la desviación típica del giróscopo y ese valor se introduce directamente en la matriz. Para $q_{5,acc}$, no conocemos la desviación típica de su medida directamente sino la del acelerómetro, por lo que debemos estimarla. Este cálculo se ha realizado mediante una simulación en la que se ha calculado en Matlab, con la función "std(v)" que calcula la desviación típica del vector de datos v. Este *array* de datos introducido ha sido $v = q_{5,acc} - q_5$, de forma que a lo que se le ha calculado la desviación típica es al ruido de la estimación de q5 por el acelerómetro. El valor obtenido fue $\sigma_{q5\_acc} = 0.313\ rad$.

$$Q = \begin{bmatrix} \sigma^2_{q5_{acc}} & 0 \\ 0 & \sigma^2_{giro} \end{bmatrix}$$ ***Ecuación 6.11***

La implementación en Simulink, que como hemos dicho es análoga a la del apartado 6.1, puede verse en el anexo G, y proporciona los siguientes resultados:

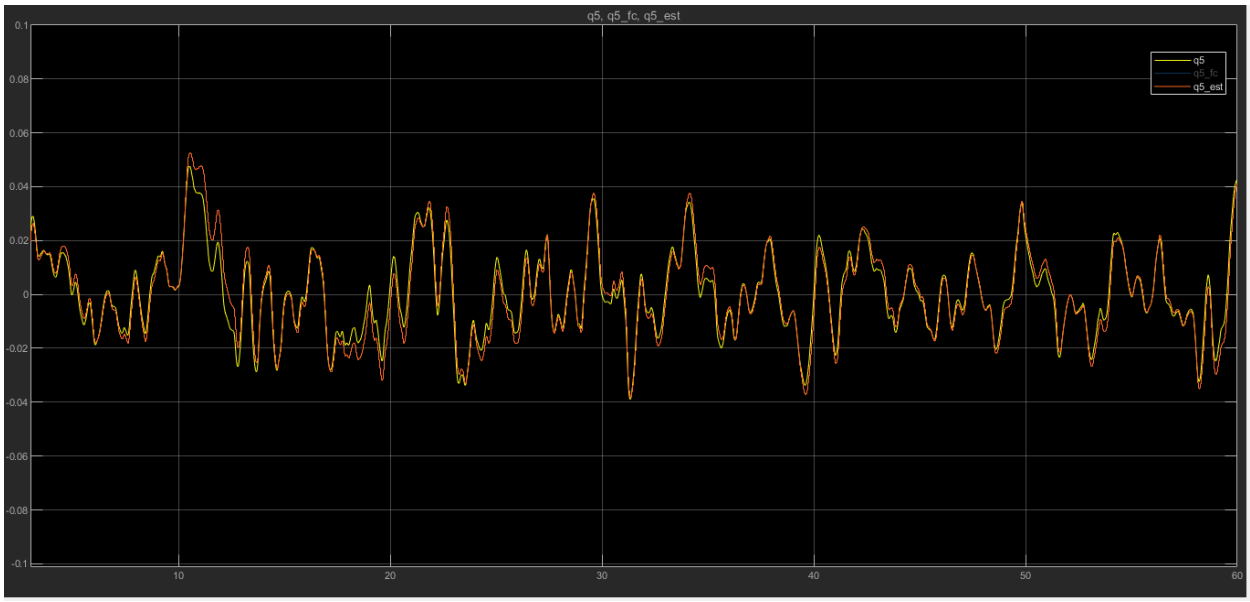


Figura 6.12 Desempeño del KF para la estimación de q5

Fuente: elaboración propia

Podemos observar, comparando los resultados de la Figura 6.11 y Figura 6.12, que el KF es bastante más eficaz

que el FC[9] por lo que a partir de este momento este último queda descartado, optando de nuevo por el filtro de Kalman para la estimación de q_5 y u_5

[9] Filtro complemetario

7 SIMULACIONES Y COMPARACIÓN ENTRE LOS MODELOS DE SENSORES

En este apartado se realizan las simulaciones finales del sistema completo y se desarrolla una discusión sobre los distintos modelos de ruido de los sensores. El sistema completo en Simulink se puede ver en el anexo G, así como el código de los ficheros .m que se utilizan para declarar todas las variables que intervienen en el problema.

7.1 Simulación 1

Comenzamos con una simulación simple, en la que no movemos el brazo manipulador, con los siguientes parámetros:

Variable	Valor inicial	Referencia	Descripción
q1	0	2 m	Posición x
q3	0	3 m	Altura z
q7	1.047 rad	1.047 rad	Posición articular 1
q8	1.25 rad	1.25 rad	Posición articular 2
sel	0		Utilizar modelo de ruido de sensores de HF + LF
Trust	0.001		Confianza en la predicción de la medida (inv. Proporcional)

Tabla 7-1 Parámetros de simulación 1

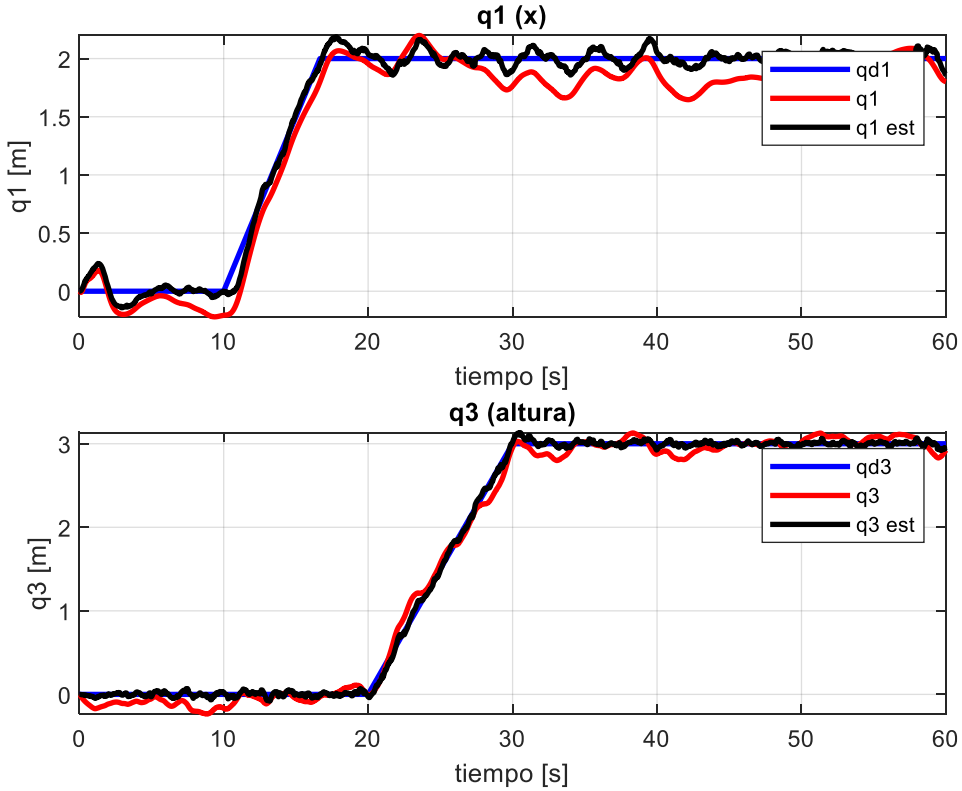


Figura 7.1 Simulación 1, q1, q3

Las variables de posición están correctamente controladas. Es natural que tengamos oscilaciones en torno a la posición de referencia, debido a la gran cantidad de inexactitudes en los sensores y demás componentes del modelo, además de que estas no son especialmente grandes. Por supuesto en la estimación de la altura tenemos menos oscilación, puesto que tenemos una estimación más exacta propiciada por dos sensores (GPS y altímetro). Además, las oscilaciones provocadas por la inercia del manipulador acoplado a la plataforma tienden a desestabilizar más en el eje horizontal que en el vertical.

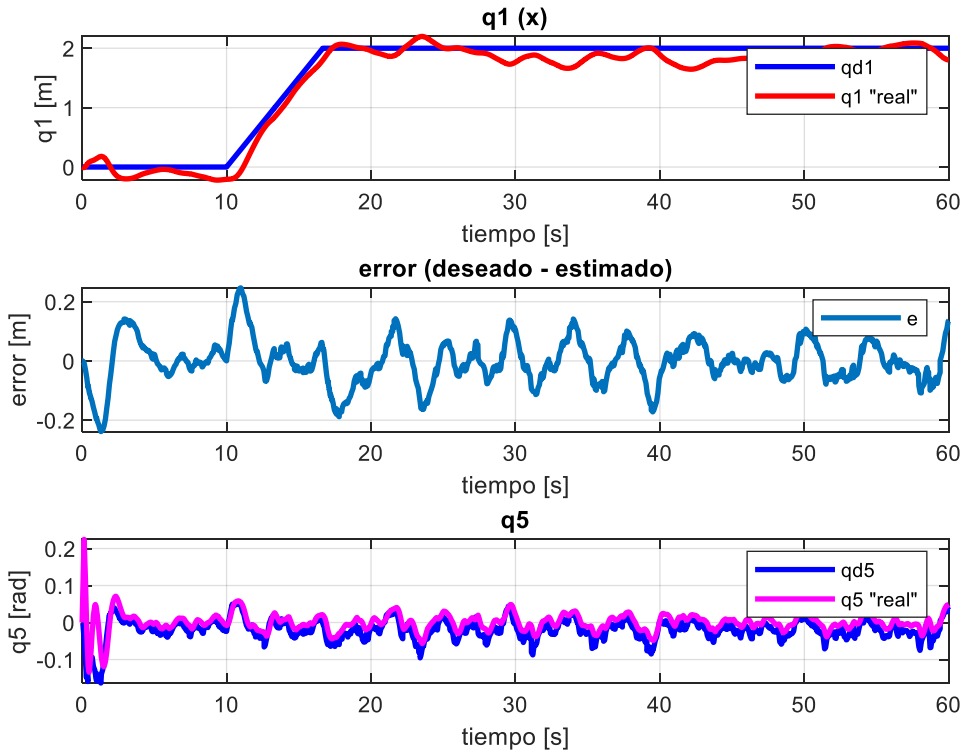


Figura 7.2 Simulación 1, q1, q5

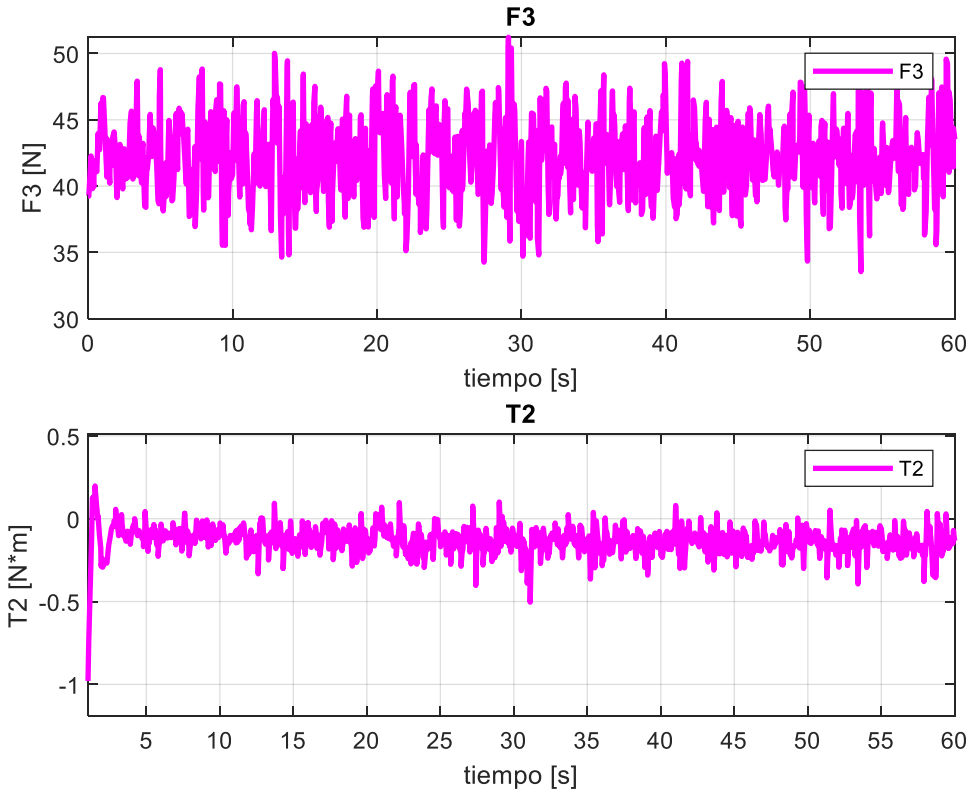


Figura 7.3 Simulación 1, F3, T2

En las dos gráficas anteriores podemos ver el comportamiento de las variables q_5, F_3 y T_2 frente a q_1. Recordamos que, para generar un movimiento horizontal, la manera de actuar era provocando un giro en q_5 y un incremento

en la fuerza vertical F_3, de forma que su proyección en el plano horizontal fuera no nula. T_2 es el par de actuación encargado de generar un giro en q_5. Estas variables, a excepción de un régimen transitorio en el par y el ángulo al que no atendemos, entran dentro del rango de normalidad. Quizás F_3 es algo más ruidosa de lo deseable, aunque los picos que presenta no son demasiado grandes y por lo tanto no estamos aplicando un control demasiado agresivo.

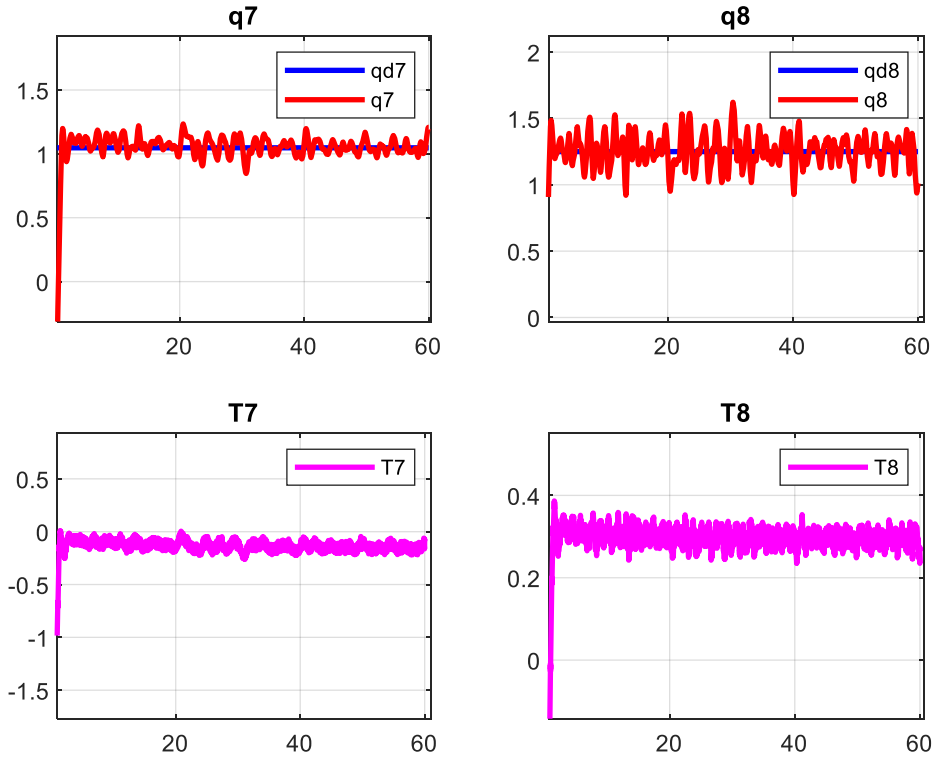


Figura 7.4 Simulación 1, q7, q8

Las variables de posición articular del brazo, q_7 y q_8 son las que más sufren las perturbaciones. Esto puede ser un indicativo de un posible campo de mejora sustancial del manipulador aéreo, puesto que intentar controlar el manipulador en posición cartesiana puede ser una tarea dificultosa con un control articular algo ruidoso.

No obstante, para los objetivos de este trabajo aceptaremos el control proporcionado en el manipulador, dejando la mejora de este como propuesta para ampliaciones futuras de este desarrollo.

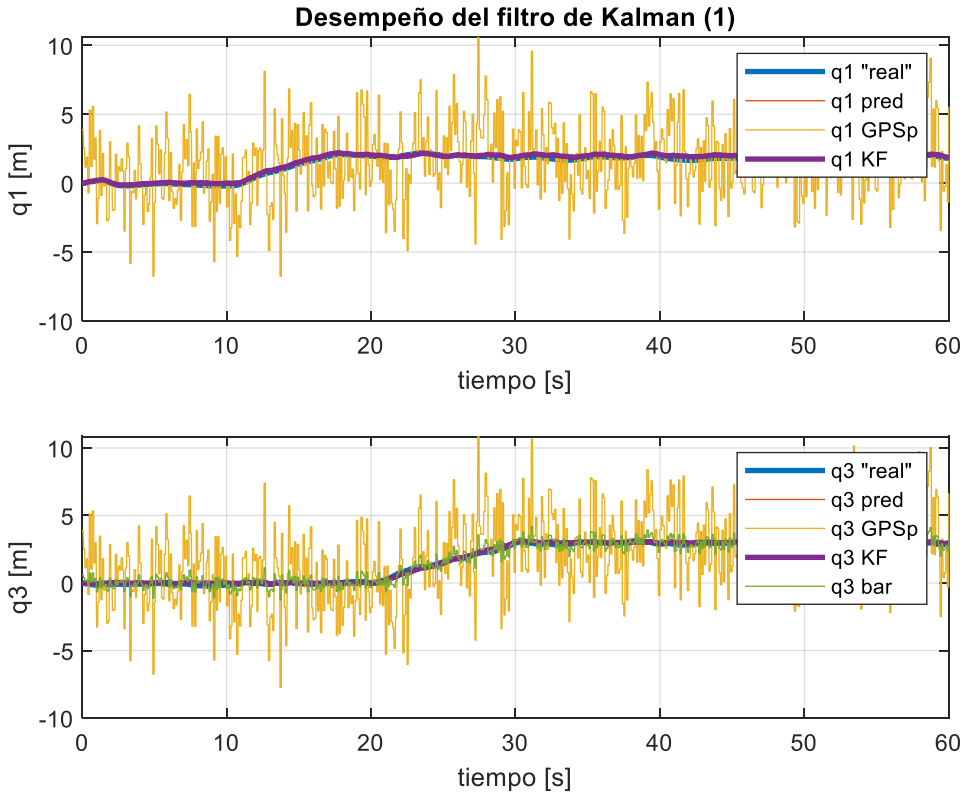


Figura 7.5 Simulación 1, KF1

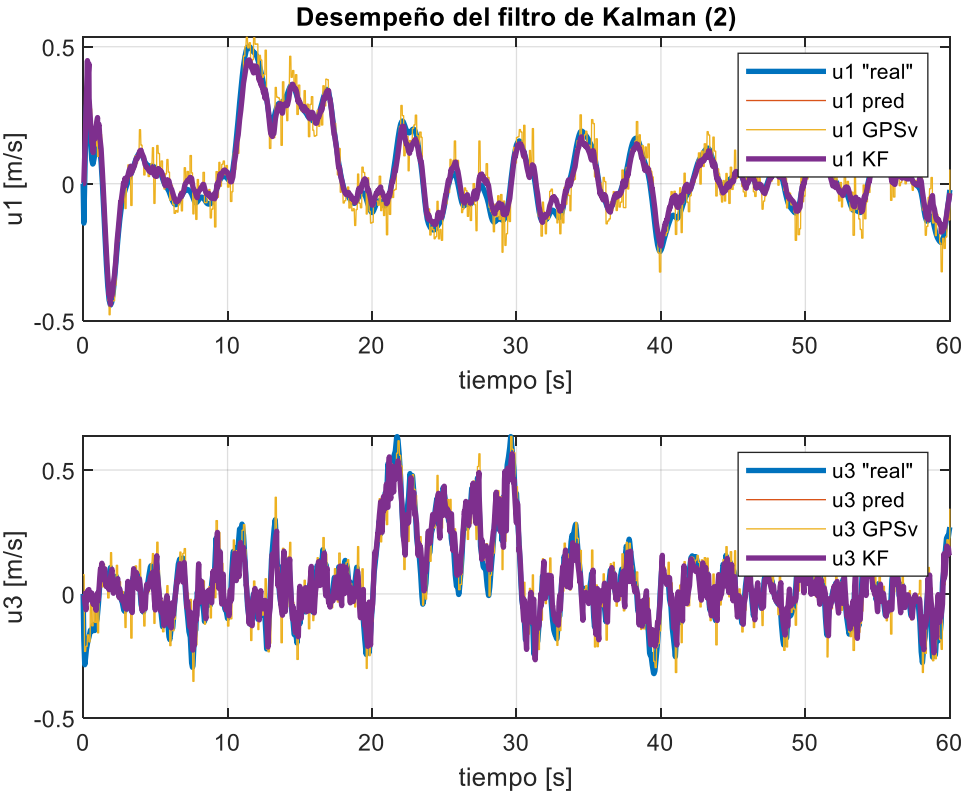


Figura 7.6 Simulación 1, KF2

El filtro de Kalman funciona correctamente. Quizás podría conseguirse una mejor estimación para las velocidades u1 y u3 si redujéramos ligeramente el parámetro trust, pero dada la eficacia en la estimación de

posición q1 y q3 no vale la pena añadir perturbación en forma de ruido para ajustar de forma ínfima la estimación de velocidad.

7.2 Simulación 2

Repetimos la simulación anterior, utilizando esta vez el modelo del ruido de los sensores del filtro de primer orden.

La tabla de parámetros de simulación es, entonces:

Variable	Valor inicial	Referencia	Descripción
q1	0	2 m	Posición x
q3	0	3 m	Altura z
q7	1.047 rad	1.047 rad	Posición articular 1
q8	1.25 rad	1.25 rad	Posición articular 2
sel	1		Utilizar modelo de ruido de sensores de filtro de primer orden
Trust	0.001		Confianza en la predicción de la medida (inv. Proporcional)

Tabla 7-2 Parámetros de simulación 2

Y realizamos la simulación.

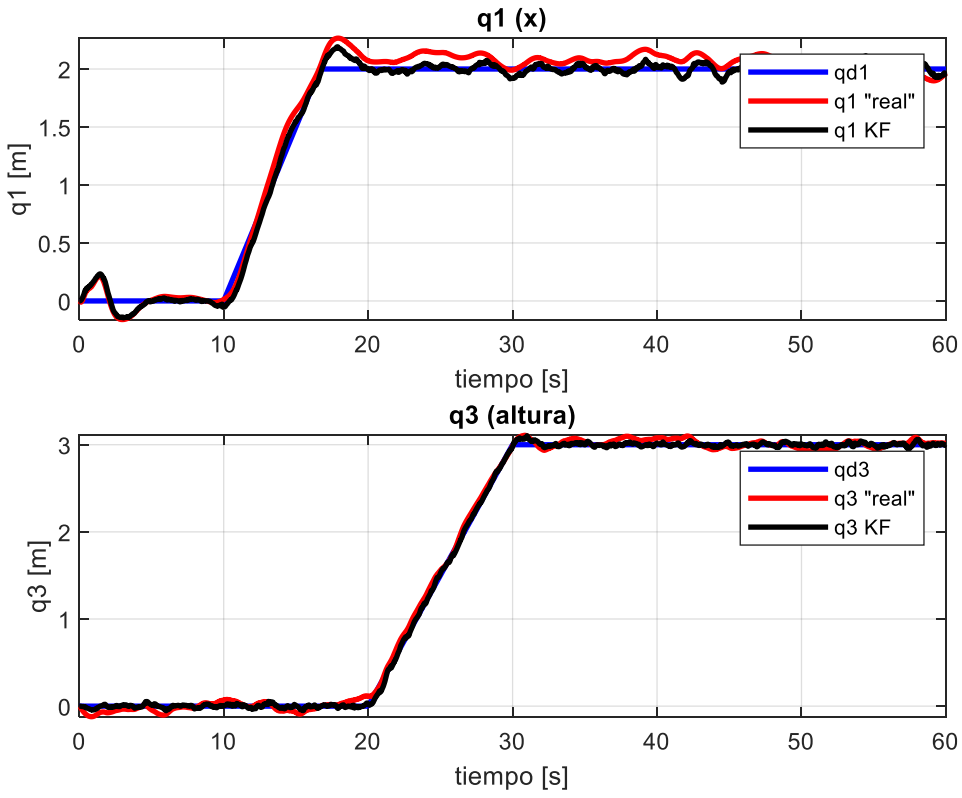


Figura 7.7 Simulación 2, q1, q3

Quizás aquí, en las gráficas de q_1 y q_3 es el único lugar en el que se hace visible alguna diferencia entre las simulaciones con los distintos modelos de ruido, especialmente en la del desplazamiento horizontal. Este comportamiento algo menos oscilatorio que en el caso de la simulación 1 aparenta haber sido propiciado por la ausencia de componente de baja frecuencia en el ruido de la medida.

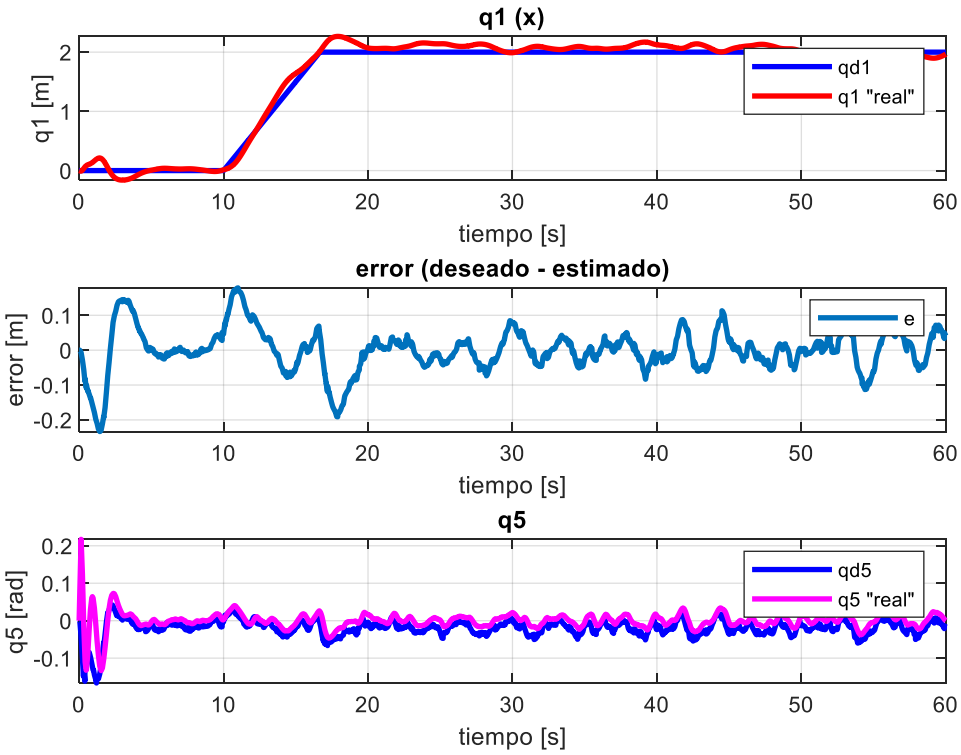


Figura 7.8 Simulación 2, q1, q5

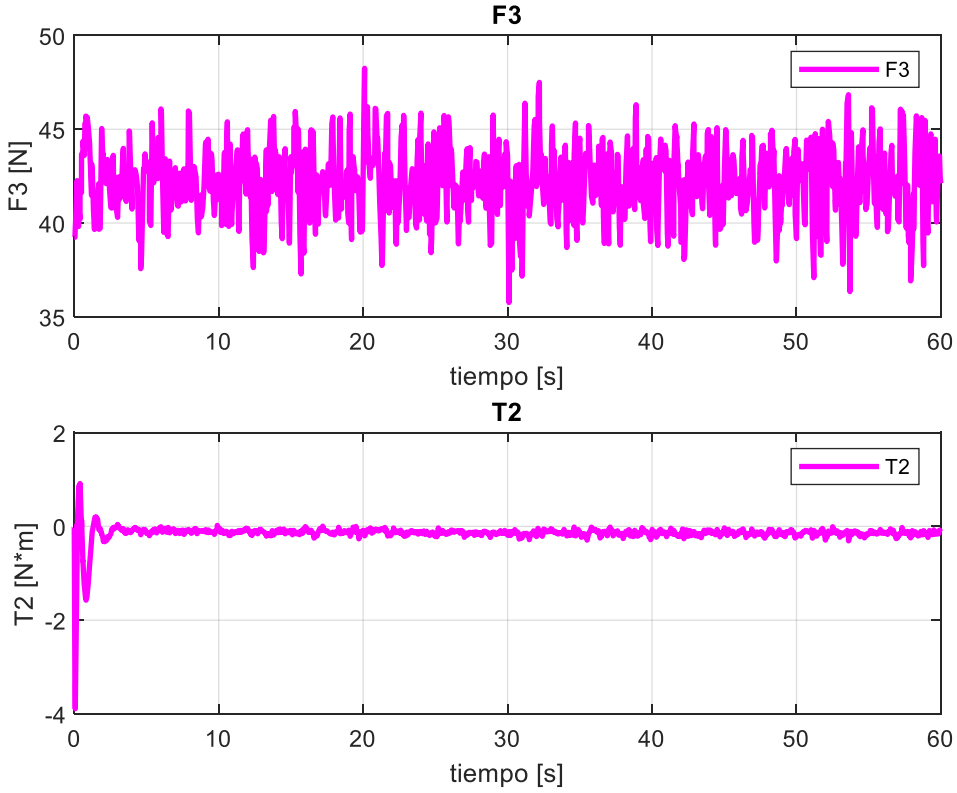


Figura 7.9 Simulación 2, F3, T2

No hay cambios que podamos identificar a simple vista en las variables de actuación F_3 ni T_2, así como tampoco se observa una dinámica distinta en q_5.

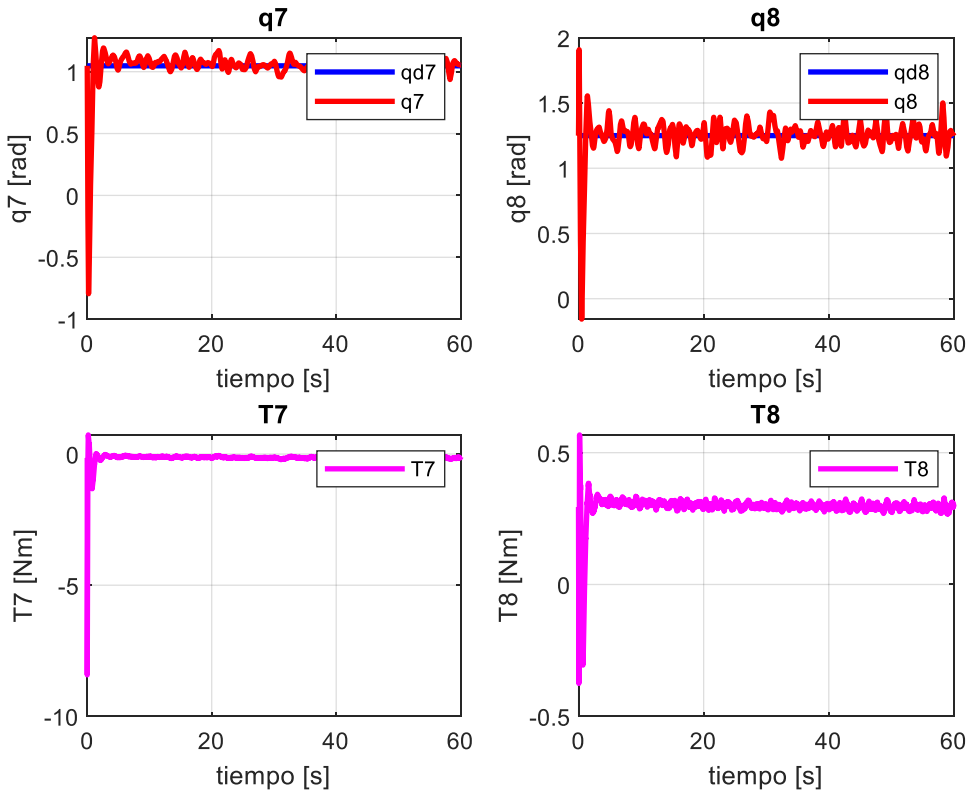


Figura 7.10 Simulación 2, q7, q8

Las variables articulares del manipulador siguen presentando también la misma dinámica que en la simulación 1.

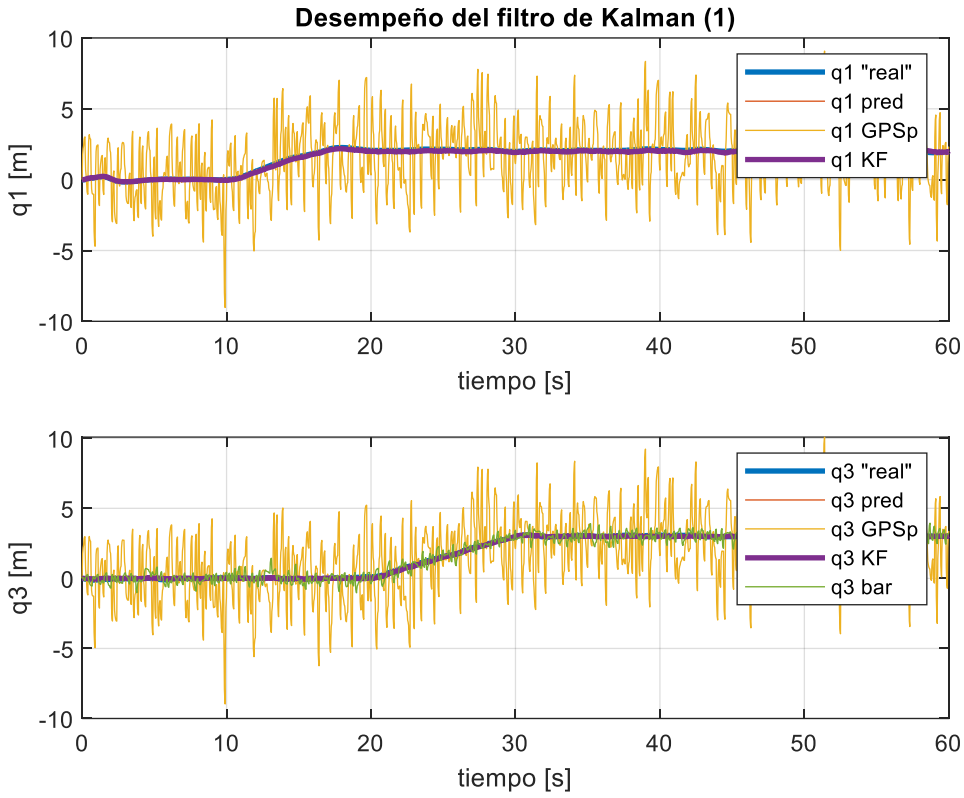


Figura 7.11 Simulación 2, KF1

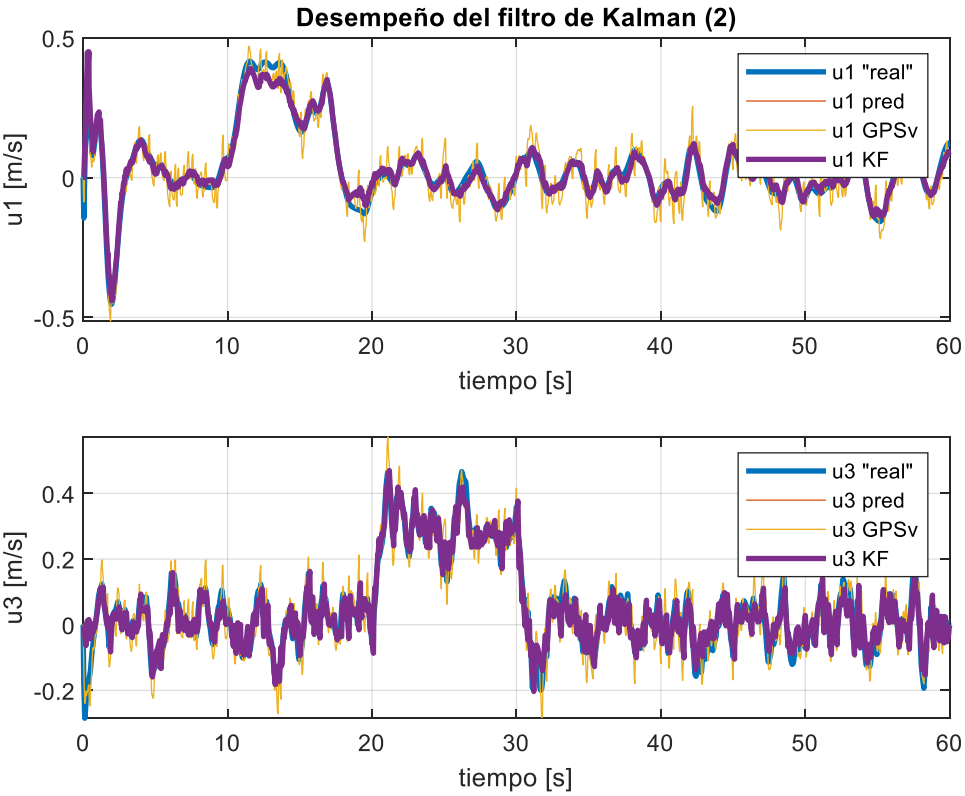


Figura 7.12 Simulación 2, KF2

7.3 Simulación 3

Para tratar de notar cambios entre ambos modelos de ruido en los sensores vamos a aumentar el parámetro "trust" de forma que el filtro corrija de forma más agresiva las estimaciones con las medidas, añadiendo así ruido al estado estimado. Además, introduciremos movimiento en el brazo manipulador, complicando así la tarea del controlador.

Variable	Valor inicial	Referencia	Descripción
q1	0	2 m	Posición x
q3	0	3 m	Altura z
q7	1.047 rad	1.8472 rad	Posición articular 1
q8	1.25 rad	1.55 rad	Posición articular 2
sel	0		Utilizar modelo de ruido de sensores de HF + LF
Trust	0.01		Confianza en la predicción de la medida (inv. Proporcional)

Tabla 7-3 Parámetros de simulación 3

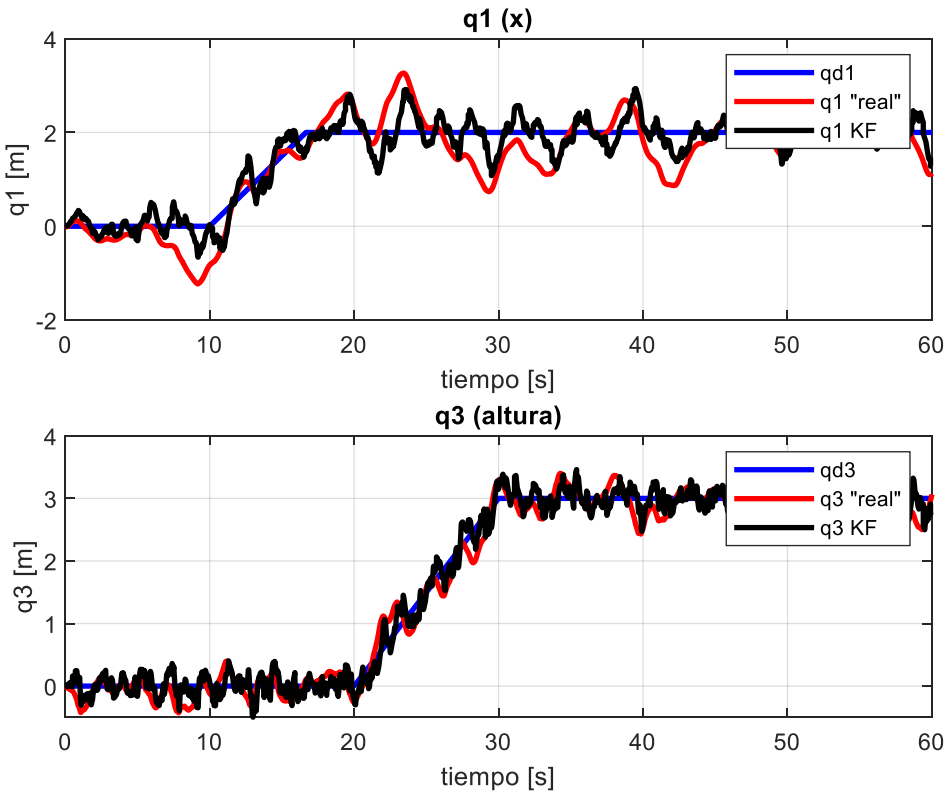


Figura 7.13 Simulación 3, q1, q3

Como cabía esperar, al tener una peor estimación de estado, también es peor el comportamiento de la plataforma frente a las consignas en posición. Llegamos a tener picos de error hasta de un metro de orden de magnitud en la posición horizontal y de medio metro en la altura. Aun así, presentando tales oscilaciones el sistema se mantiene "estable" en torno a un punto de equilibrio.

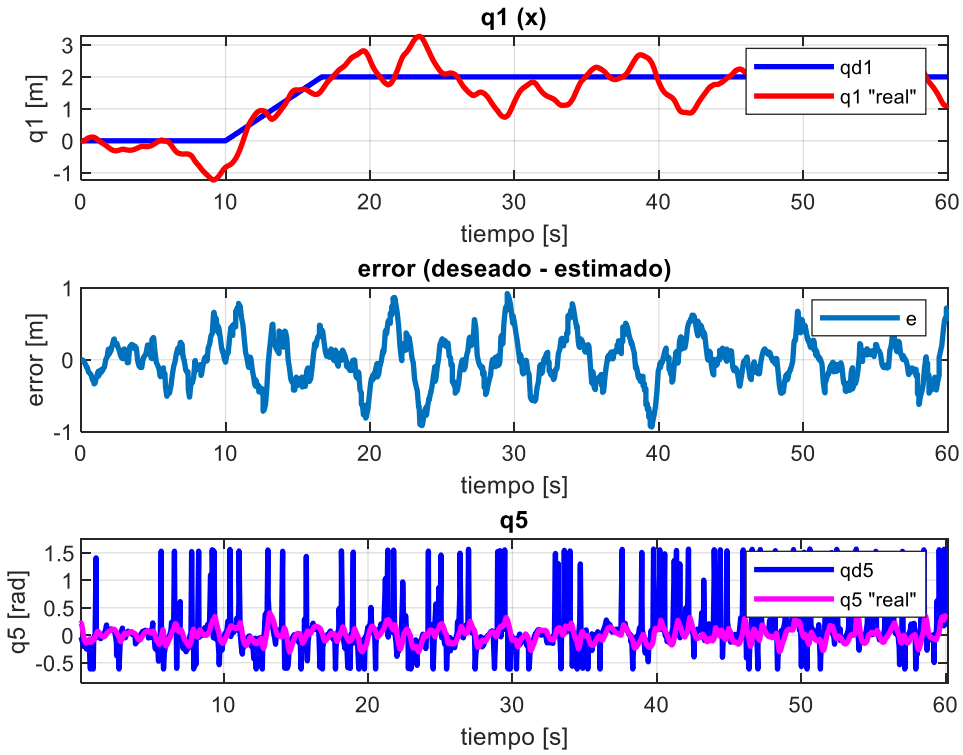


Figura 7.14 Simulación 3, q1, q5

En la figura anterior se puede observar cómo el controlador comienza a actuar de una forma agresiva, llegando a unos picos de ángulo de *pitch* de referencia muy grandes de más de un radián. Aunque estos ángulos de referencia no lleguen a alcanzarse, haciendo zoom se puede observar que q_5 llega a ser incluso del orden de 0.4 radianes (aproximadamente 23 grados), lo que empieza a ser un comportamiento peligroso.

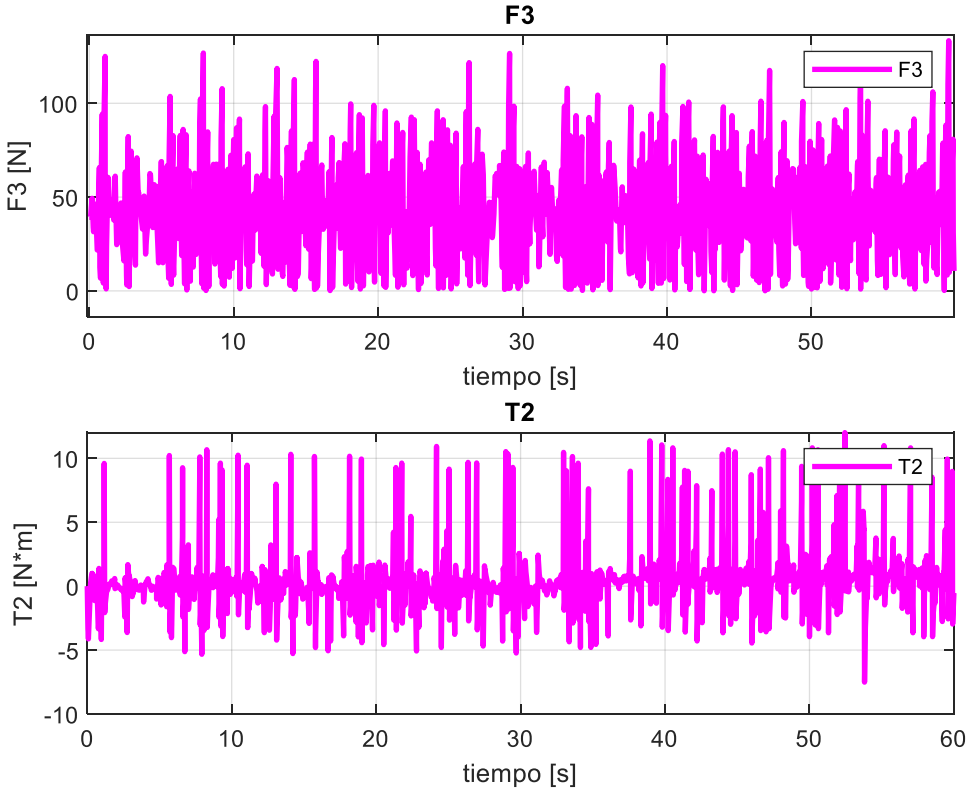


Figura 7.15 Simulación 3, F3, T2

También actúan de forma agresiva las señales de salida del controlador, F_3 y T_2.

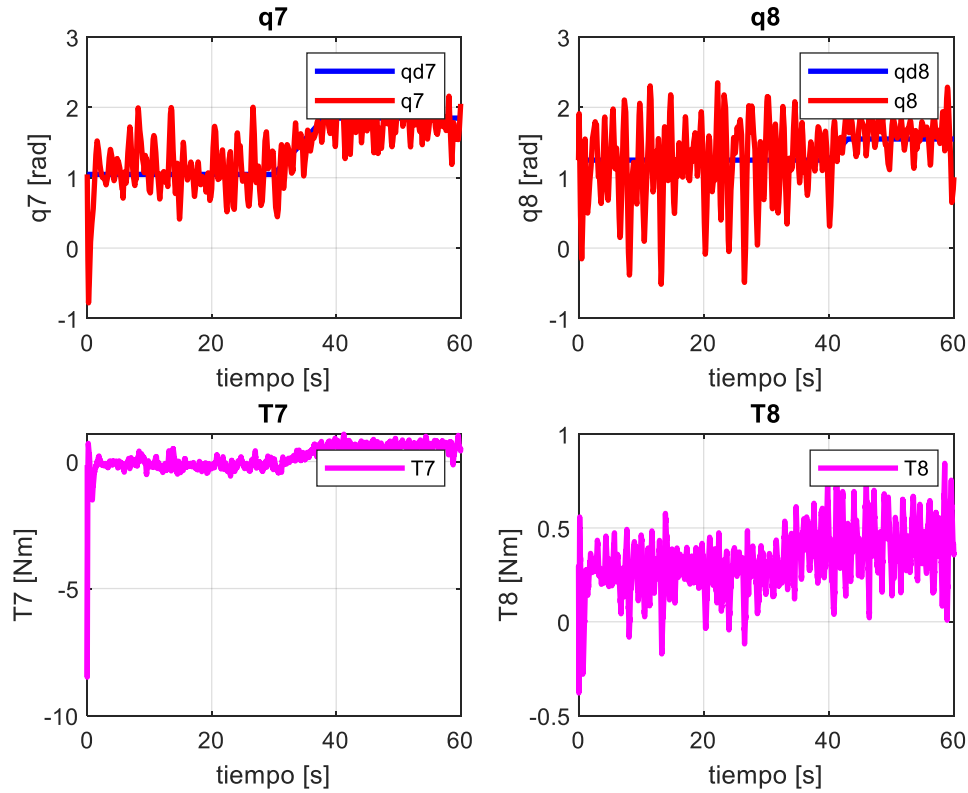


Figura 7.16 Simulación 3, q7, q8

Observando las variables articulares del brazo y sus oscilaciones inadmisibles que se están produciendo

principalmente por el movimiento en la plataforma, podemos afirmar que la elección de un valor de "trust" como la de este apartado es inviable de cara a una aplicación real de un manipulador aéreo.

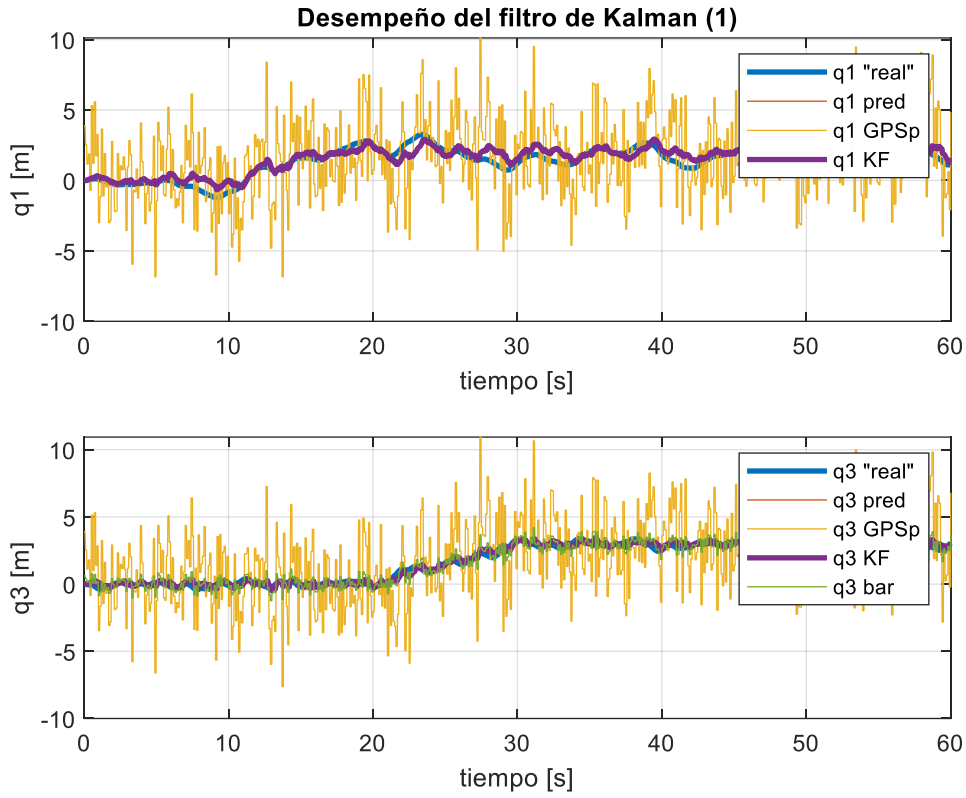


Figura 7.17 Simulación 3, KF1

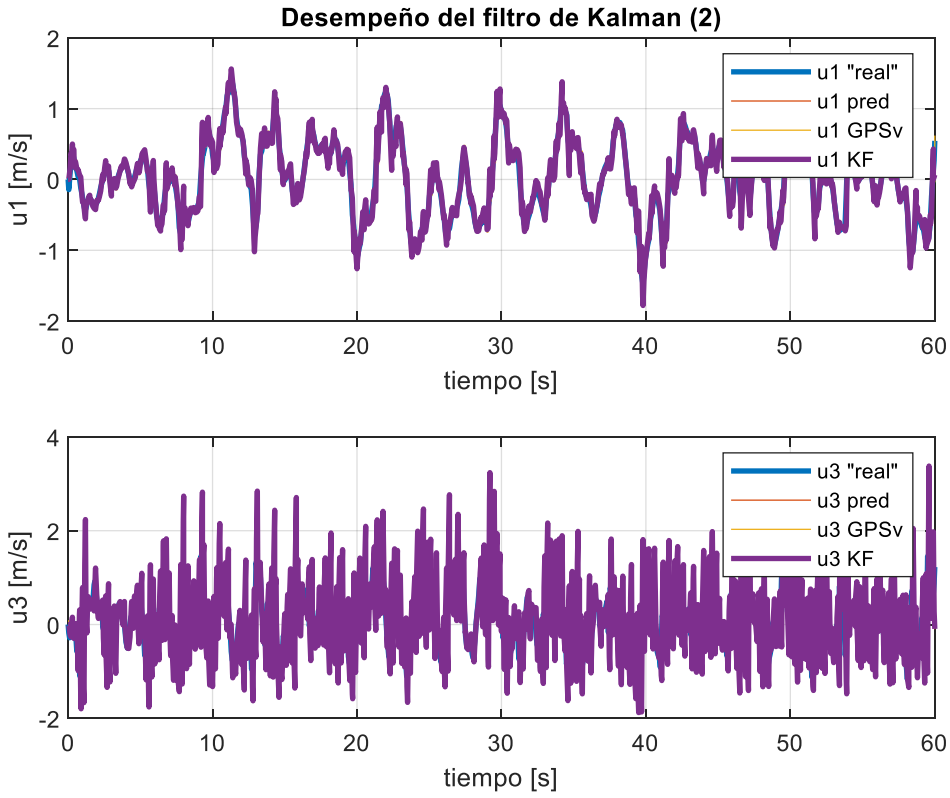


Figura 7.18 Simulación 3, KF2

En las dos gráficas anteriores se muestra la estimación de estado a la que lleva una elección del parámetro “trust” de 0.01.

7.4 Simulación 4

Comparamos los resultados de la simulación anterior con los que se obtendrían mediante el modelo de ruido con filtro de primer orden, manteniendo el parámetro trust en 0.01.

Variable	Valor inicial	Referencia	Descripción
q1	0	2 m	Posición x
q3	0	3 m	Altura z
q7	1.047 rad	1.8472 rad	Posición articular 1
q8	1.25 rad	1.55 rad	Posición articular 2
sel	1		Utilizar modelo de ruido de sensores de filtro de primer orden
Trust	0.01		Confianza en la predicción de la medida (inv. Proporcional)

Tabla 7-4 Parámetros de simulación 4

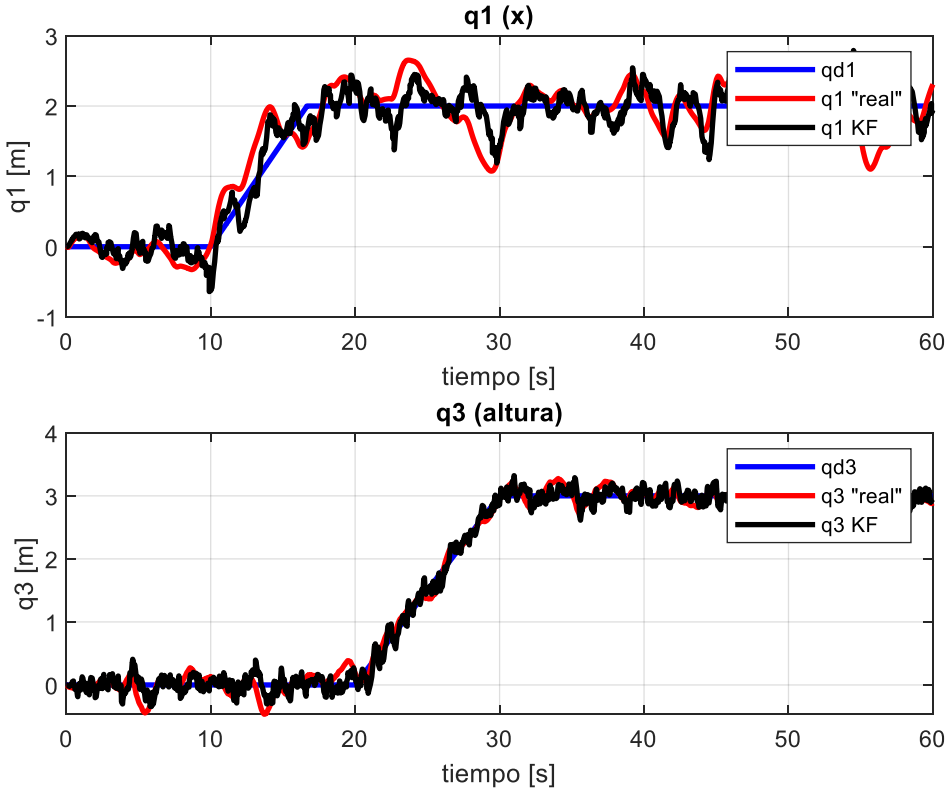


Figura 7.19 Simulación 4, q1, q3

De nuevo se observa que este modelado del ruido produce algo menos perturbaciones en la posición del UAV para la misma desviación típica, gracias a la forma menos cuadrada de su evolución temporal y a la inexistencia de perturbaciones de baja frecuencia.

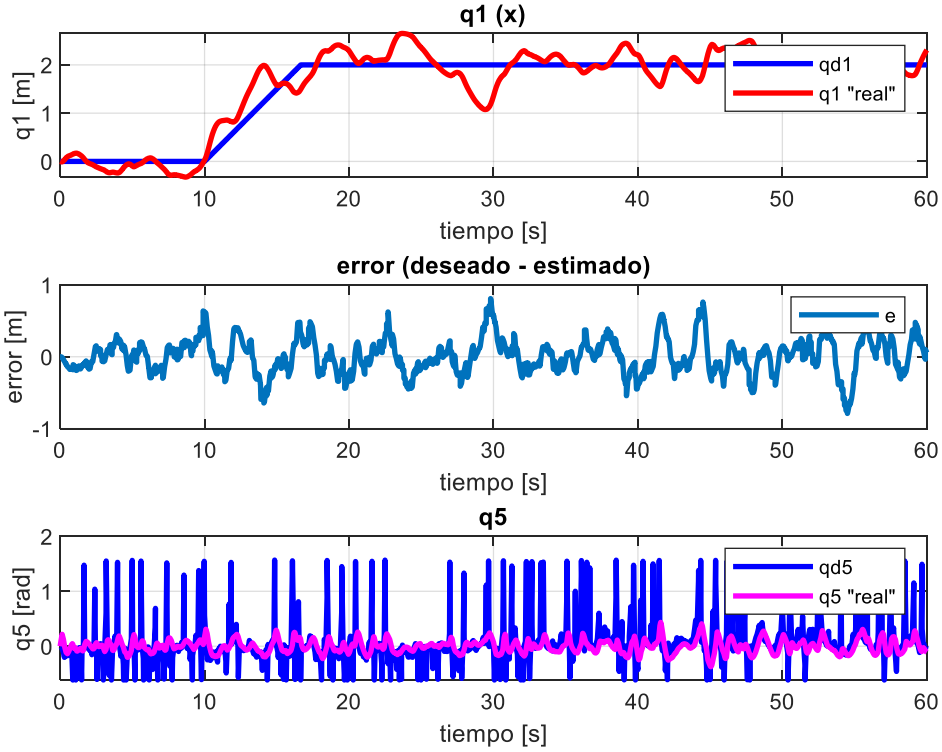


Figura 7.20 Simulación 4, q1, q5

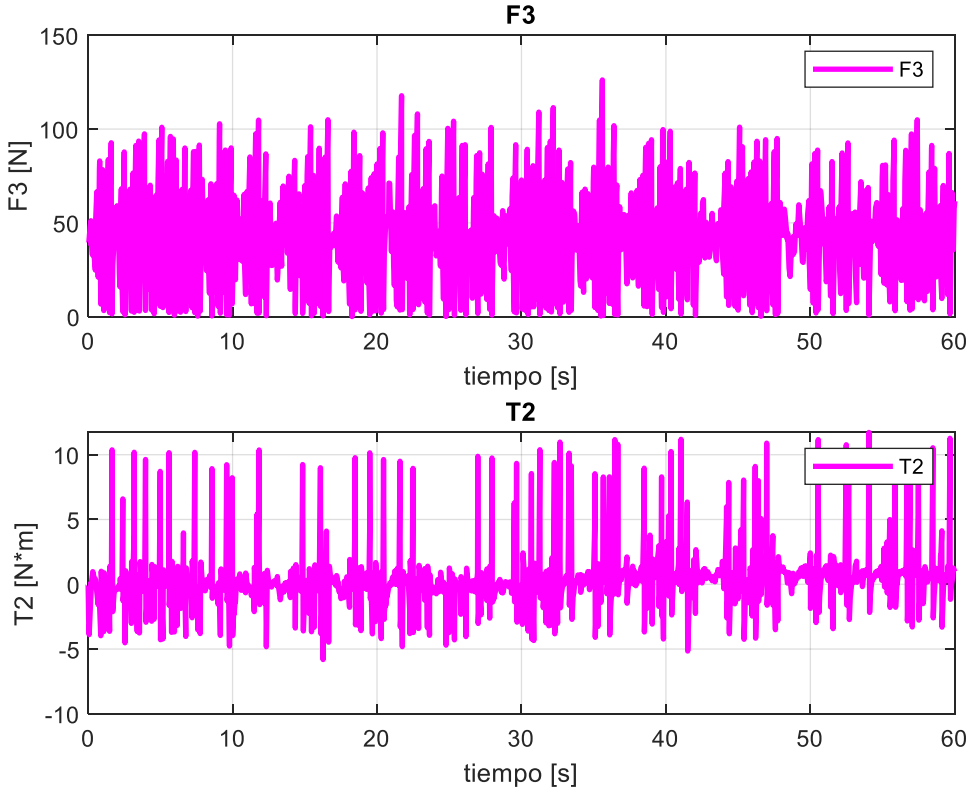


Figura 7.21 Simulación 4, F3, T2

El control de posición sigue actuando de una forma brusca, al igual que lo hacía con el modelo de HF + LF, por lo que no podemos concluir que este comportamiento del controlador sea propiciado por ninguno de los dos

métodos de simular las señales de medida.

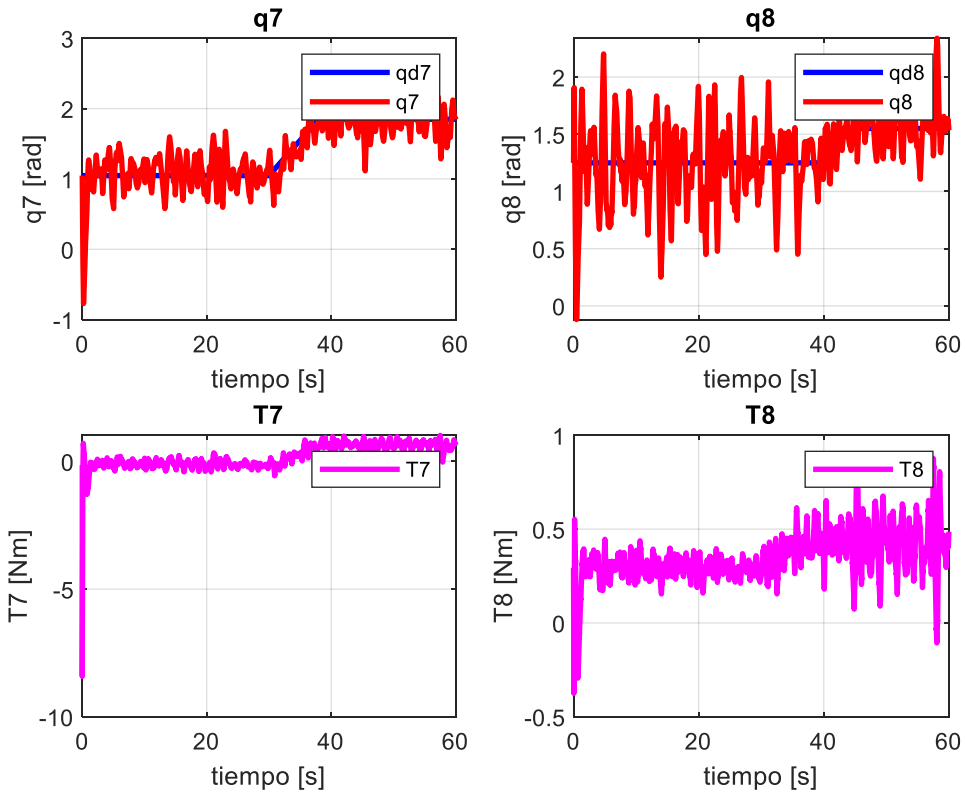


Figura 7.22 Simulación 4, q7, q8

Sí ocurre que en el control del manipulador se mejora sustancialmente la evolución de q_7, en un grado tal que podríamos pensar que la generación artificial de las medidas de los codificadores ópticos se ve beneficiada por este modelo del ruido. Si vemos la señal de T_7, que tiene bastante poca oscilación, podemos deducir que esta mejora en q7 se debe a que en la derivación de la medida de los codificadores se genera menos ruido cuando se utiliza el modelo de filtro de primer orden.

No obstante, q_8 sigue completamente fuera de control, por lo que necesitaríamos de otra estrategia de control (o una mejor estimación de estado que no generara tanto ruido) si pretendiéramos aplicar estos desarrollos en la realidad. En definitiva, para un buen comportamiento del manipulador aéreo se necesita una elección del parámetro "trust" de al menos 0.001, como en las simulaciones 1 y 2.

Por último, podemos ver la estimación de estado del filtro de Kalman en esta simulación:

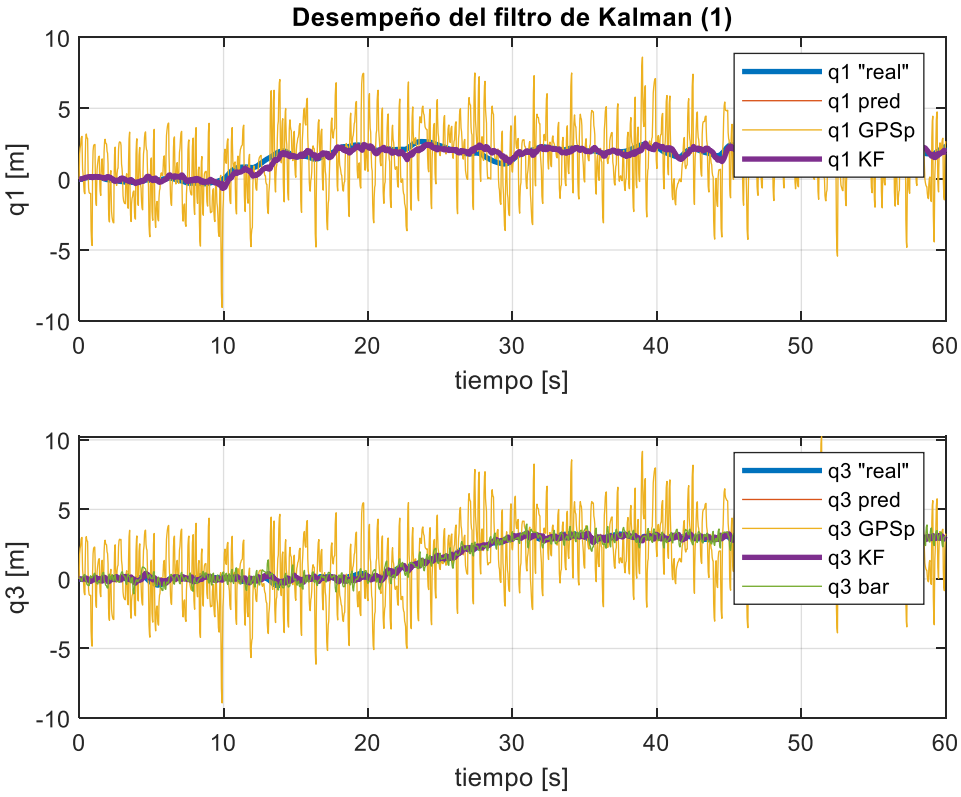


Figura 7.23 Simulación 4, KF1

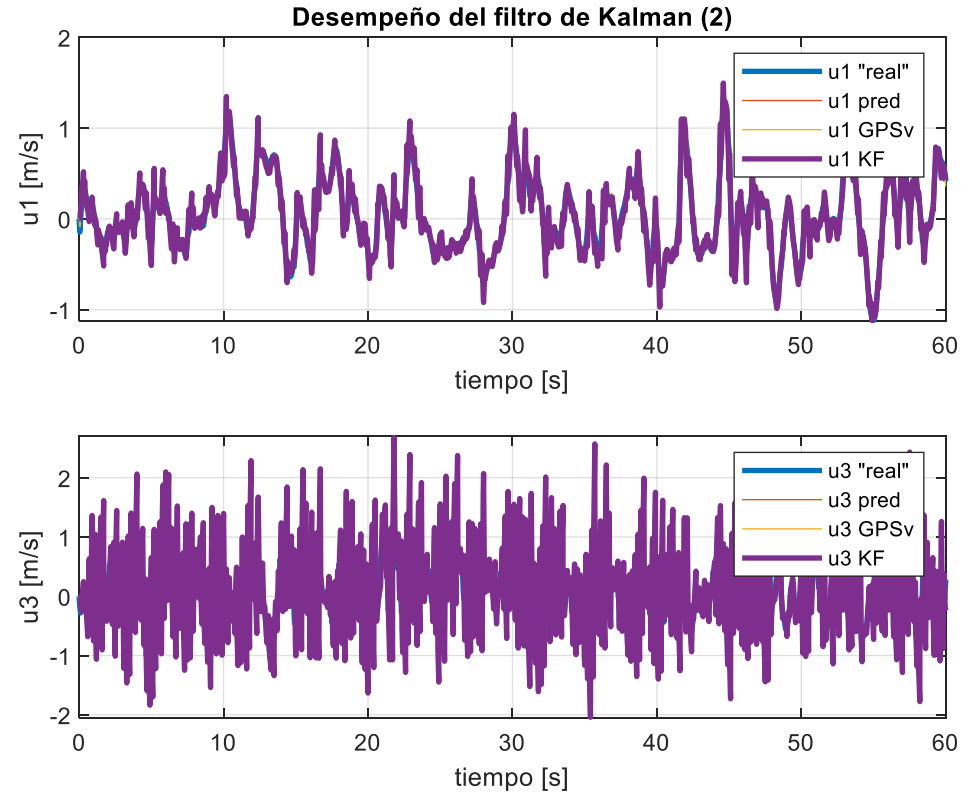


Figura 7.24 Simulación 4, KF2

7.5 Comparación de simulaciones 1 a 4

Se han realizado cuatro simulaciones tratando de encontrar discrepancias en el comportamiento del sistema cuando se utilizan dos modelos para el ruido de los sensores. Debido a la existencia de la etapa de estimación de estado mediante el filtro de Kalman, estas discrepancias se hacen algo menos visibles, hecho que nos obliga a modificar el valor de "trust" para obtener unos resultados u otros. Si quisiéramos seguir aumentando estas difierencias, tendríamos que seguir aumentando el parámetro de confianza en la predicción en nuestras simulaciones, acción que queda descartada porque se llega a inestabilizar el sistema si se reduce un orden de magnitud más de lo que lo hemos hecho. Esto significa que, no habiendo encontrado diferencias de comportamiento en ninguna de las simulaciones realizadas, concluimos en que el modelo de ruido, siempre que se ajuste dentro de unos márgenes normales no afecta significativamente al comportamiento del sistema cuando existe etapa de estimación de estado. Además, gracias a estas simulaciones se ha manifestado el hecho de que en una aplicación real no se puede aumentar demasiado el parámetro trust, y la gran importancia que tiene su estimación (lo cual se puede hacer para este sistema).

Por otro lado, el desarrollo de dos modelos distintos para el ruido de los sensores sigue estando justificado en el hecho de que han sido creados con dos filosofías distintas: el de HF + LF con un objetivo más realista y suponiendo que se puede obtener una descripción estadística de los sensores algo detallada por medio de experimentación-caracterización El otro modelo, del filtro de primer orden, contempla menos parámetros y un planteamiento más adecuado para simulaciones o aplicaciones en las que se se tengan los parámetros de los sensores reales a partir de *datasheets*. También vale la pena contemplar este modelo cuando quieran realizarse simulaciones en las que intervengan señales con ruido aleatorio pero las cuales tengan un papel menos protagonista.

7.6 Simulación 5

El objetivo de esta simulación es comparar el comportamiento del manipulador aéreo equipado con sensorización real con respecto al caso ideal de la sección 4.3. Para ello, se mantiene la plataforma en vuelo fijo (*hovering*) y se realiza un movimiento del brazo, en este caso manteniendo la articulación 8 fija en su posición inicial y moviendo únicamente el primer *link*[10] del manipulador.

Variable	Valor inicial	Referencia	Descripción
q1	0	0	Posición x
q3	0	0	Altura z
q7	1.047 rad	2.618 rad	Posición articular 1
q8	1.25 rad	1.25 rad	Posición articular 2
sel	1		Utilizar modelo de ruido de sensores de filtro de primer orden
Trust	0.001		Confianza en la predicción de la medida (inv. Proporcional)

Tabla 7-5 Parámetros de simulación 5

Los resultados de simulación son los siguientes:

[10] Vínculo

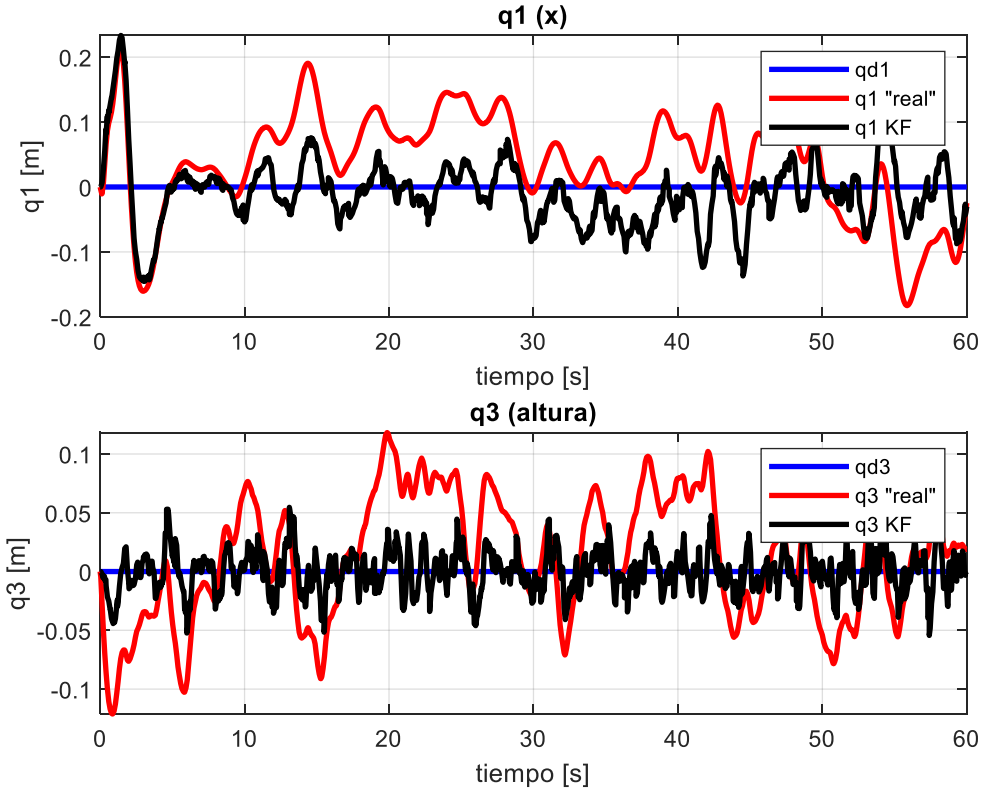


Figura 7.25 Simulación 5, q1, q3

Las posiciones cartesianas se mantienen en el punto de vuelo, con un error menor de 20 cm en el caso de la posición horizontal, y de 10 cm en el caso de la altura.

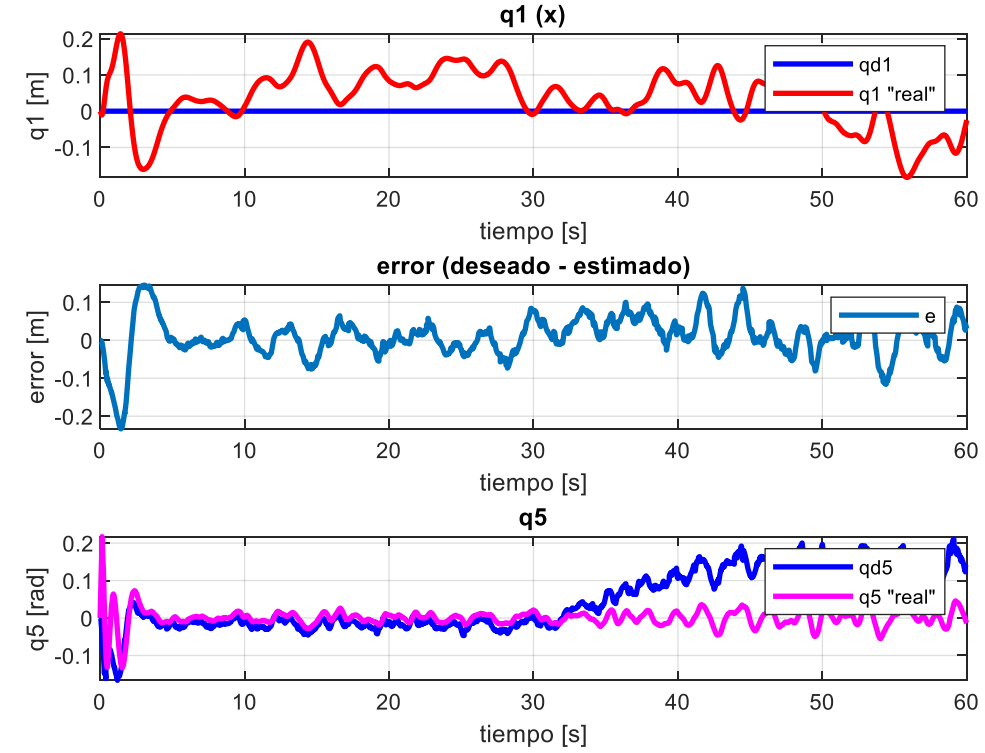


Figura 7.26 Simulación 5, q1, q5

Ahora sí se puede ver claramente la discrepancia entre las q_5 real y deseada relacionadas con que la única forma que tiene el controlador de generar un par T_2 para compensar el momento que produce el levantamiento del brazo es creando una q_5d ficticia, como se explicó en 4.3.2. Podemos ver dicho par en la siguiente gráfica:

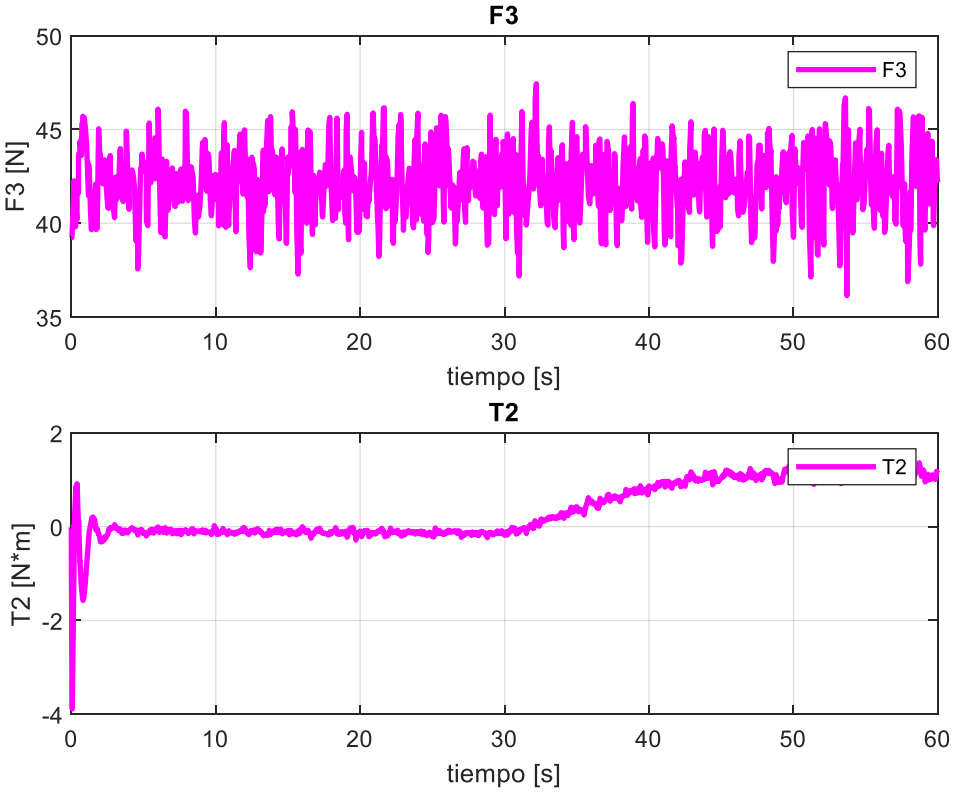


Figura 7.27 Simulación 5, F3, T2

Las posiciones articulares son las siguientes:

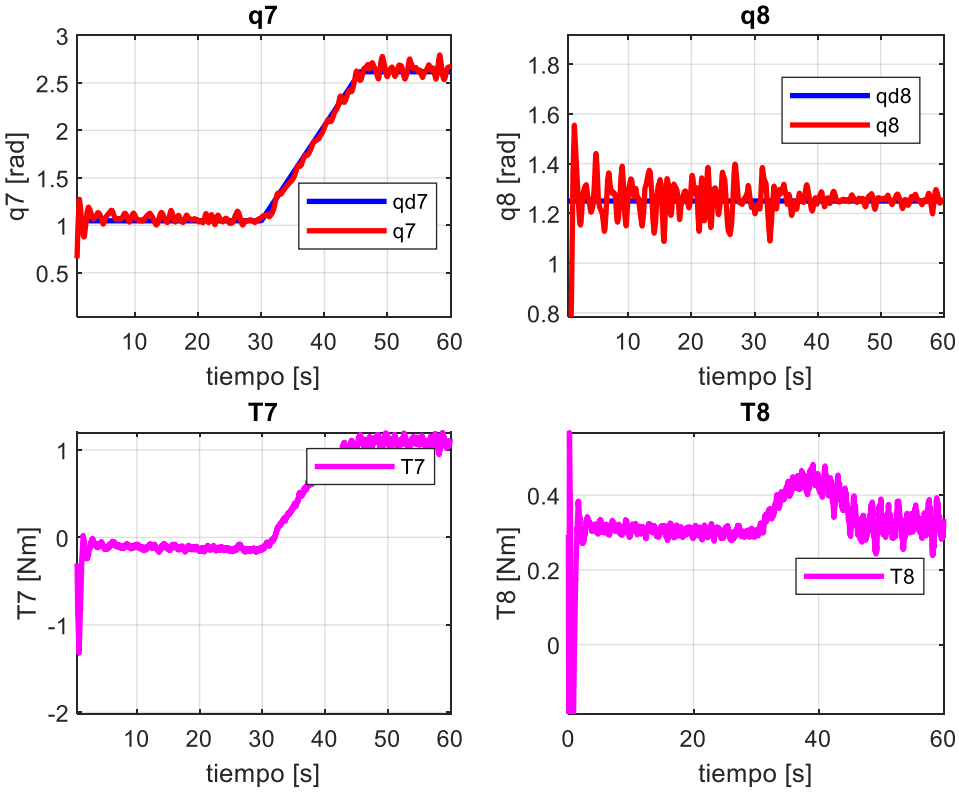


Figura 7.28 Simulación 5, q7, q8

Se alcanza cómodamente la referencia de q_7, aunque en q_8 aún se tienen algunas perturbaciones notables (de unos 10 grados) que deben ser tomadas en cuenta para la utilización del manipulador en alguna aplicación real.

Por último, las estimaciones del filtro de Kalman.

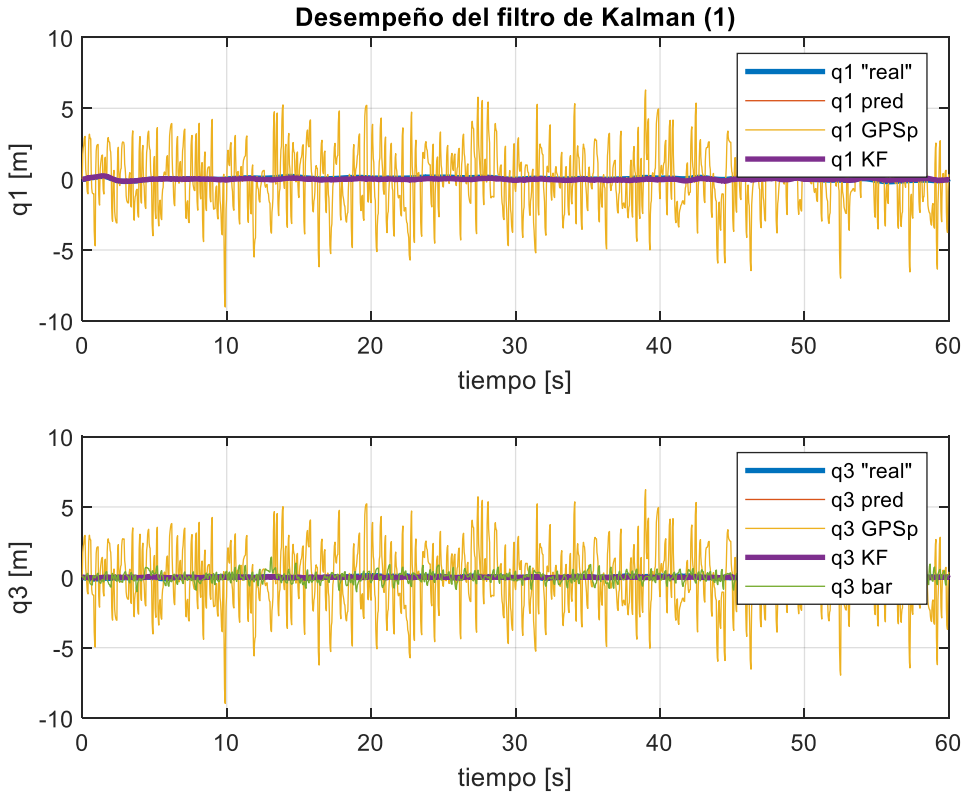


Figura 7.29 Simulación 5, KF1

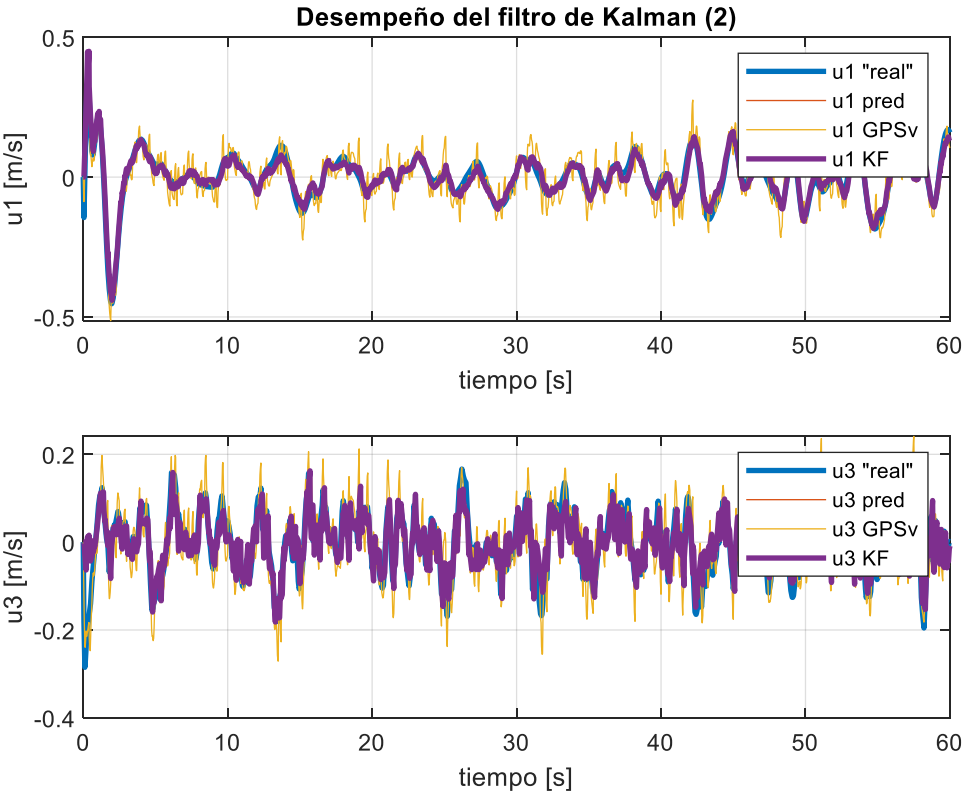


Figura 7.30 Simulación 5, KF2

En resumen, el comportamiento observado del sistema con sensorización real es muy parecido al caso en el que no se contemplaba la sensorización, únicamente añadiendo las oscilaciones propias de la adición del ruido. En

cuanto a las posiciones cartesianas se demuestra la robustez del controlador desarrollado a partir de la inversión del modelo del UAV junto a la etapa de estimación de estado, ya que se mantiene al UAV en torno a la posición de referencia con unos errores menores de 20 cm, teniendo en cuenta que la medida de posición horizontal por parte del GPSp tiene un error de 2.5m. Para el control del manipulador se ha demostrado que se requiere un mayor grado de exactitud, dadas las oscilaciones que se presentan en los vínculos rotativos, principalmente en el segundo, que es en el que más se amplifican los movimientos de la plataforma y el primer eslabón del brazo. Quizás un controlador que considerara las dinámicas del manipulador aéreo completo y no solo del manipuador o una fase de estimación de estado para suavizar las medidas de los codificadores ópticos serían los siguientes pasos a seguir para conseguir un comportamiento mejor en una aplicación real.

8 CONCLUSIONES Y TRABAJO FUTURO

8.1 Conclusiones

Los manipuladores aéreos son sistemas muy complejos y se necesita entrar en muchas ramas de la ingeniería para completar su desarrollo; en este TFG se ha abordado el estudio completo de estos elementos. Desde el desarrollo de las ecuaciones de la física hasta el tratamiento de la implementación realista con el modelado de sensores. Pasando por la robótica tanto para la plataforma o quadrotor como para el brazo robótico, sin olvidar la ingeniería de control necesaria para hacer que todo funcione. El modelado en Simulink ha resultado ser una herramienta muy cómoda para la simulación de nuestros sistemas, permitiéndonos crear todas las piezas del puzle de forma modular y progresivamente.

Estas simulaciones nos han permitido ver cómo es el funcionamiento de un robot de estas características y aprender mucho sobre ello. También para ilustrar otros factores que a simple vista pasan desapercibidos como, por ejemplo, la precisión que se necesita para que un sistema con unas dinámicas tan rápidas e inestables como lo es un quadrotor sea capaz de realizar tareas de manipulación de forma eficaz.

La inclusión de modelos realistas de los sensores ha sido otro pilar fundamental de este trabajo. Esto se hace nesario porque lo que se ha desarrollado en esta memoria es un primer paso de acercamiento hacia la experimentación y la posterior implantación. Cuando se trata con sistemas robóticos reales en lugar de con modelos matemáticos, la sensorización es un factor fundamental puesto que es la forma que tiene el sistema de situarse, requisito fundamental para poder desempeñar su función. Se han planteado dos métodos para emular los sensores típicos de los robots autónomos, con dos filosofías distintas que responden tanto a la etapa de simulación como a la de experimentación.

Y con ello se ha demostrado también la trascendental importancia que tienen los algoritmos de estimación de estado como el filtro de Kalman implementado en este trabajo. Esta importancia reside en el hecho de que se pueden abaratar significativamente los precios de muchos sistemas, acercándolos un paso más a la comercialización gracias a que con una buena estimación se pueden conseguir resultados robustos aun disponiendo de unos sensores baratos o no demasiado precisos.

Desde una perspectiva global, en este trabajo se ha desarrollado una herramienta que permite trabajar en las primeras etapas de manipuladores aéreos de forma sencilla. Este entorno de trabajo desarrollado permite probar de una forma segura y muy poco costosa distintas configuraciones de sensores, filtros, controladores y demás aspectos relacionados con los manipuladores aéreos que de otra forma no sería posible, como hemos podido observar en las simulaciones por la peligrosidad de disponer de un sistema real mal ajustado.

Los manipuladores aéreos son sistemas súmamente versátiles y de una gran utilidad para muchísimos campos de aplicación, especialmente el industrial. Es por eso que se justifica la gran investigación que se está llevando a cabo, desde trabajos de iniciación como este hasta la vanguardia más innovadora en las universidades. Sin duda son ya una realidad, y en pocos años veremos cómo revolucionan en muchos aspectos a nuestra industria.

8.2 Trabajo futuro

Si bien se ha desarrollado un trabajo completo atendiendo a casi todas las etapas en el modelado de un manipulador aéreo, existen puntos de mejora y de ampliaciones futuras, entre los que podemos destacar los más importantes:

- Experimentación con manipulador aéreo real para caracterización de parámetros y comparación de comportamiento frente al simulado.
- Extensión al caso 3D.

- Modelado preciso de las constantes dinámicas de los cuerpos.
- Control avanzado del manipulador, incluyendo control de coordenadas cartesianas y mejora del desempeño de las articulaciones.
- Conexión con controladores reales como PX4.
- Modelo de animación para representación visual e intuitiva del comportamiento del sistema.
- Modelo de actuación en motores propulsores reales para la obtención de las fuerzas y pares de actuación a partir de velocidad de giro y comportamiento aerodinámico.
- Modelo energético sobre el consumo de baterías.

Gracias.

REFERENCIAS

[1] K. Kondak, M. Bernard, N. Meyer and G. Hommel, "Autonomously Flying VTOL-Robots: Modeling and Control", *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Roma, 2007, pp. 736-741.

[2] L.A. Sandino, M. Bejar, M. Kondak and A. Ollero, "Advances in Modeling and Control of Tethered Unmanned Helicopters to Enhance Hovering Performance", *Journal of Intelligent and Robotic Systems*, vol. 73, pp. 3-18, Jan. 2014.

[3] L. Sandino, "Modeling and control techniques of autonomous helicopters for landing on moving platforms", PhD. Thesis, Dept. ingeniería de sistemas y automática, Univ. Sevilla, 2016. Accessed on Dec. 20, 2019. [Online]. Available at https://www.researchgate.net/publication/303784513_Modeling_and_control_techniques_of_autonomous_helicopters_for_landing_on_moving_platforms

[4] A. Ollero, G. Heredia, A. Franchi, G. Antonelli, K. Kondak, A. Sanfeliu, A. Viguria, J.R. Martinez de Dios, F. Pierri, J. Cortes, A. Sntamaria-Navarro, M.A. Trujillo Soto, R. Balachandran, J. Andrade-Cetto, A. Rodriguez Castaño, "The AEROARMS Project: Aerial Robots with Advanced Manipulation Capabilities for Inspection and Maintenance", *IEEE Robotics & Automation Magazine*, vol. 25, pp. 12-23, 2018.

[5] H. Lee, H. Kim Kim, H. y H. Jim Kim, "Planning and Control for Collision-Free Cooperative Aerial Transportation", *IEEE Transactions on Automation Science and Engineering,* vol. 15, pp 189-201, 2016

[6] S. Kim, S. Choi, H. Lee, H. Jim Kim, "Vision-based collaborative lifting using quadrotor UAVs", *14th International Conference on Control, Automation and Systems (ICCAS 2014)*, Seúl, 2014, pp. 1169-1174.

[7] A. Barrientos, L.F. Peñín, C. Balaguer and R. Aracil, Cinemática del robot in *Fundamentos de robótica*, 2. Madrid: McGraw-Hill, 2007, Ch. 4, pp.119-214.

[8] J.J. Craig, *Introduction to robotics: Mechanics and Control*, 3rd ed. United States: Pearson Education, Inc. 2005.

[9] A. Ollero Baturone (2015). "Aeroarms". [Online]. Available at: https://aeroarms-project.eu/ [Accessed 12 Aug. 2019].

[10] Prodrone (2015). "Revolutionary Drones for Professionals: Prodrone". [Online]. Available at: https://www.prodrone.com/ [Accessed 20 Aug. 2019].

[11] DLR (2012). “DLR: Institute of Robotics and Mechatronics”. [Online]. Available at: https://www.dlr.de/rm/en/desktopdefault.aspx/tabid-8017 [Accessed 1 Sep. 2019].

[12] J. Redolfi, D. Alejandro, “Filtro complementario para estimacion de actitud aplicado al controlador embebido de un cuatrirrotor”, *Congreso Argentino de Sistemas Embebidos*, Argentina, 2011.

[13] S. Thrun, W. Burgard and D. Fox, Gaussian Filters in Probabilistic Robotics, 1. United States: MIT Press, 2005, Ch.3, pp. 33-66.

[14] Inven Sense, “High Performance 6-Axis MEMS MotionTracking™ Device” ICM-20602 datasheet, 2016 [Revised Mar. 2016].

[15] u-blox, “NEO-M8 u-blox M8 concurrent GNSS modules”, NEO-M8 datasheet, 2016 [Revised Aug. 2016].

[16] Faulhaber, “Encoder”, IER-1024 datasheet, 2017 [Revised 2017].

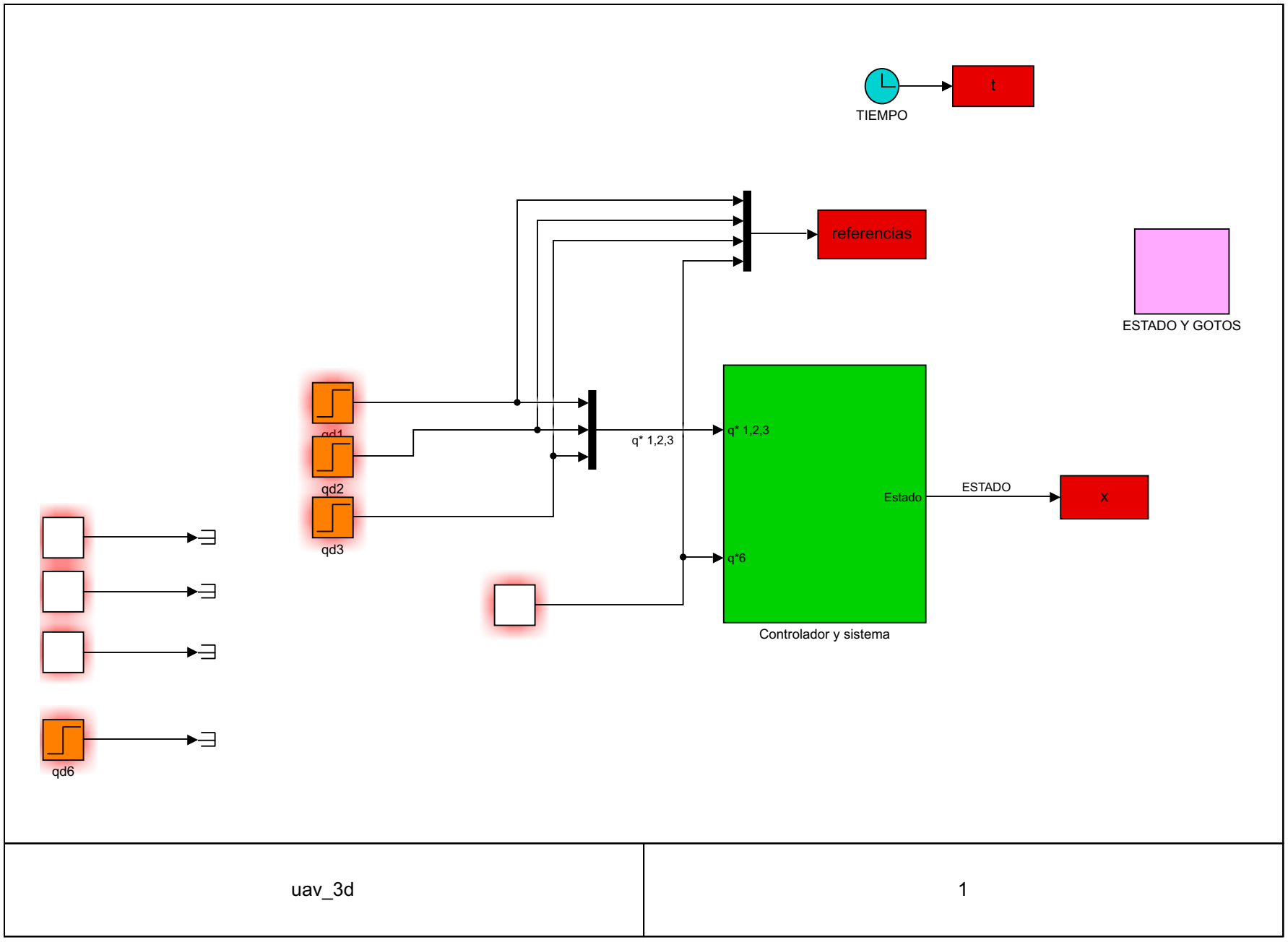
[17] Bosch, “BMP180 Digital pressure sensor”, BMP180 datasheet, 2013 [Revised 2013].

[18] K. Baizid, F. Caccavale, S. Chiaverini, G. Giglio and F. Pierri, "Safety in coordinated control of multiple unmanned aerial vehicle manipulator systems: Case of obstacle avoidance," *22nd Mediterranean Conference on Control and Automation*, Palermo, 2014, pp. 1299-1304.

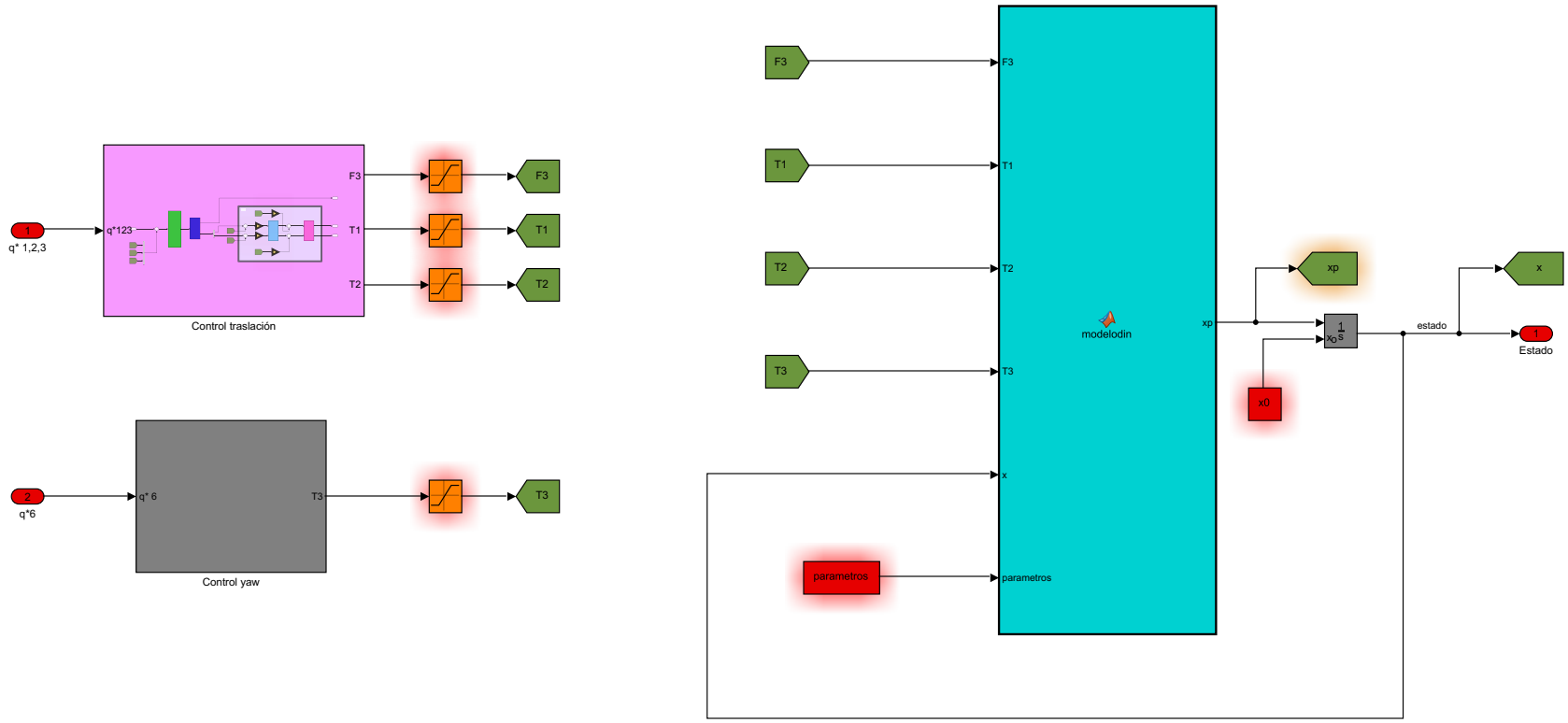
[19] J. A. Millan-Romera, H. Perez-Leon, A. Castillejo-Calle, I. Maza and A. Ollero, "ROS-MAGNA, a ROS-based framework for the definition and management of multi-UAS cooperative missions," *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, Atlanta, GA, USA, 2019, pp. 1477-1486.

[20] LBA Industrial Mining and Cía S. de R.L. de C.V. (2012). “ABM Industrial”. [Online]. Available at: http://www.abm-industrial.com/ [Accessed 11 Nov. 2019].

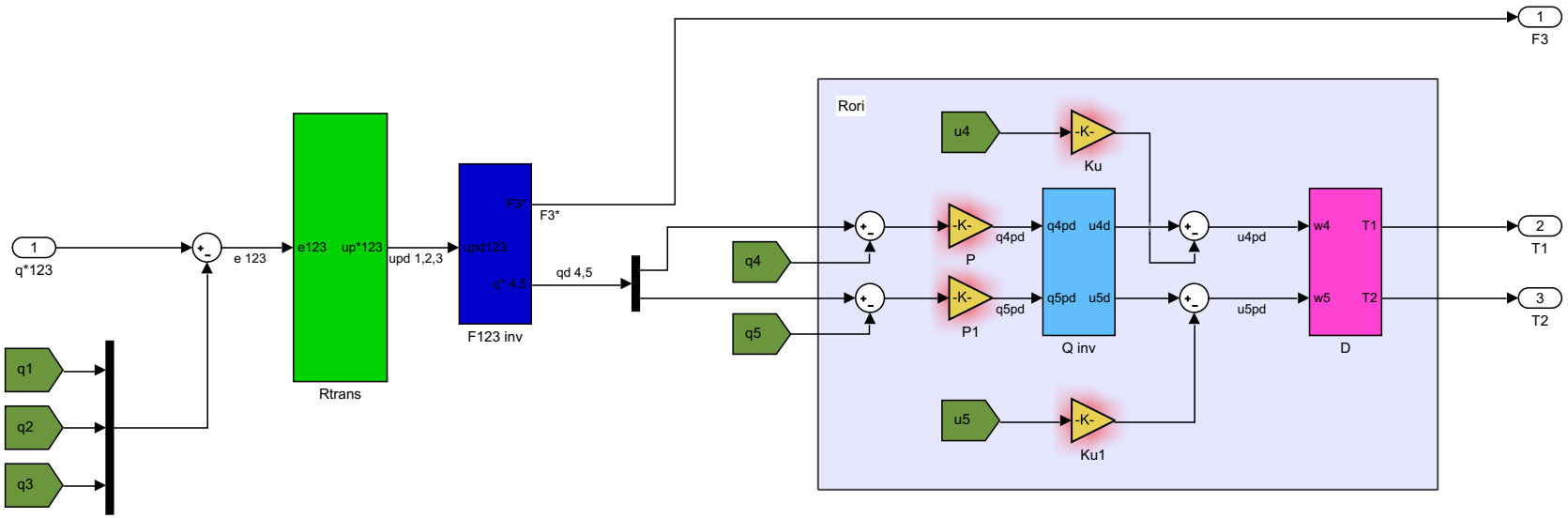
ANEXO A

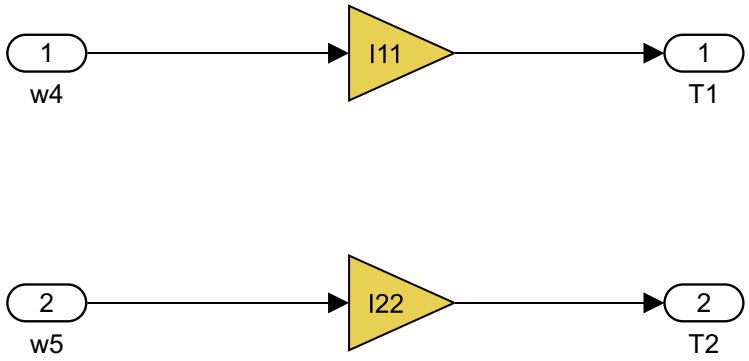

TIEMPO
t
referencias
ESTADO Y GOTOS
qd1
qd2
qd3
q* 1,2,3
q* 1,2,3
q*6
Estado
ESTADO
X
Controlador y sistema
qd6

uav_3d	1

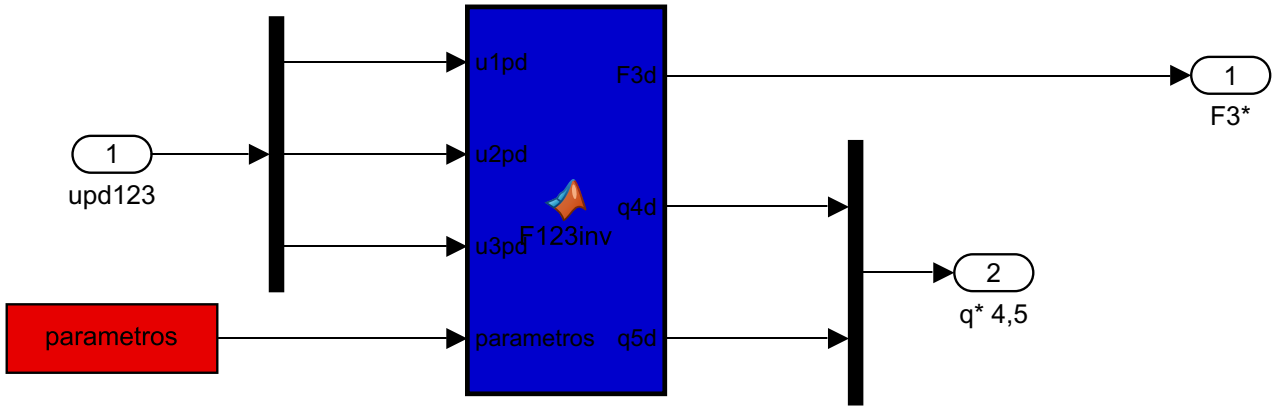


uav_3d/Controlador y sistema	2


1
F3
Rori
u4
-K-
Ku
1
q*123
e 123
e123
up*123
upd 1,2,3
upd123
q* 4,5
F3*
F3*
F123 inv
qd 4,5
Rtrans
q4
q5
-K-
P
q4pd
q4pd
u4d
u4pd
w4
T1
2
T1
-K-
P1
q5pd
q5pd
u5d
u5pd
w5
T2
3
T2
Q inv
D
u5
-K-
Ku1
q1
q2
q3

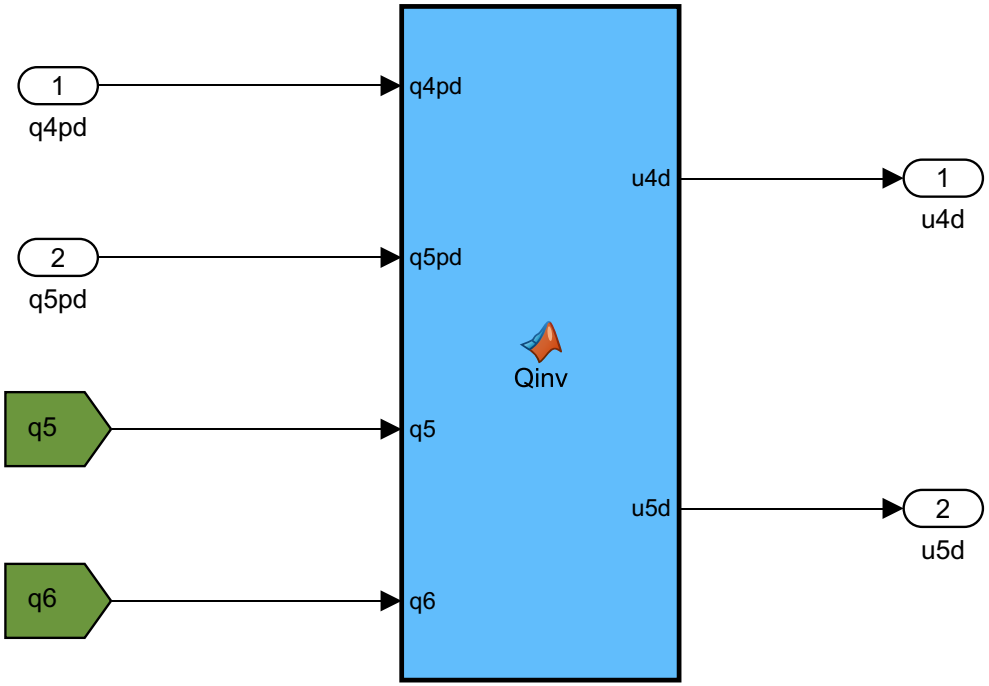


uav_3d/Controlador y sistema/Control traslación/D	4



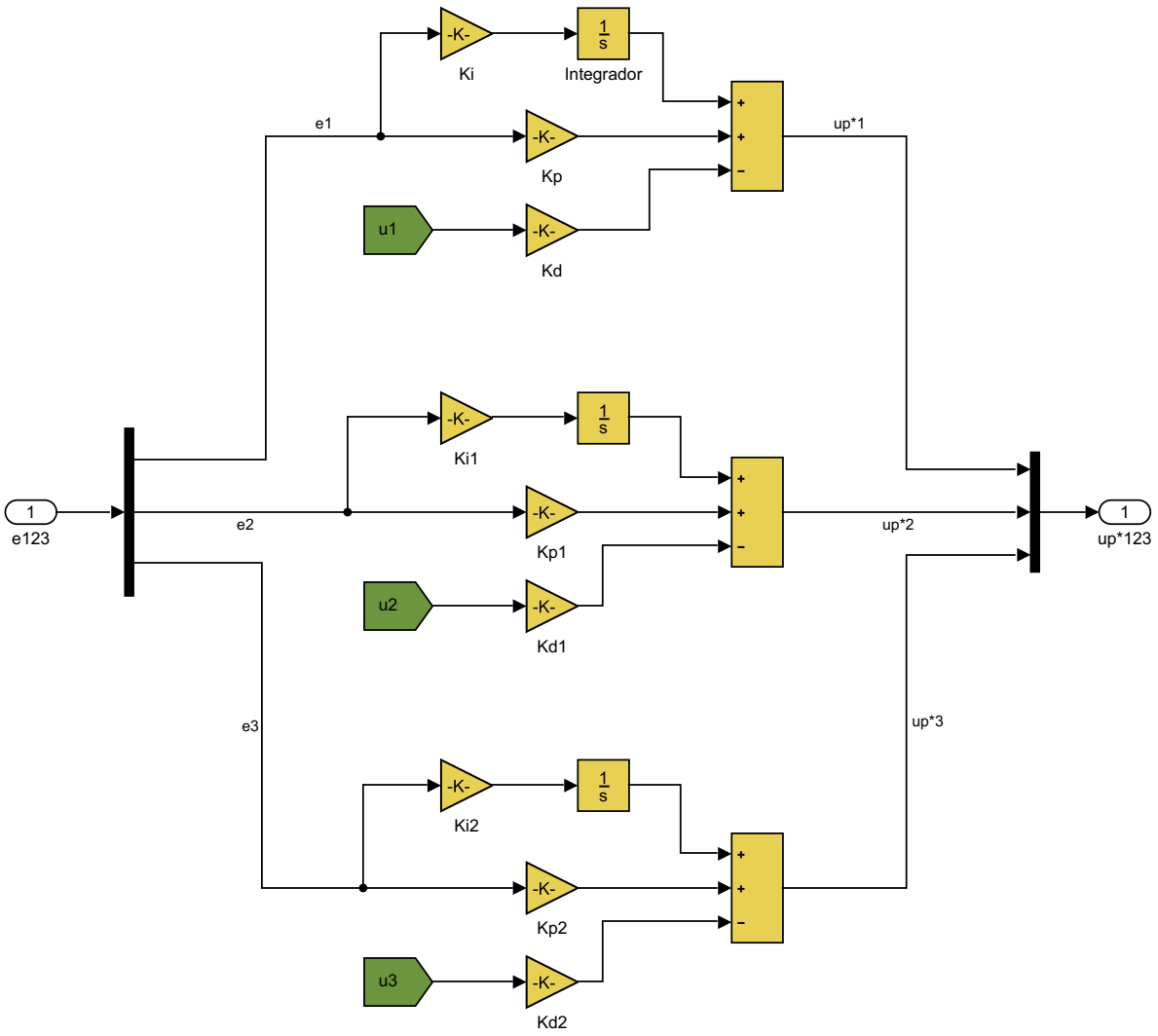
uav_3d/Controlador y sistema/Control traslación/F123 inv	5

```
function [F3d,q4d,q5d] = F123inv(u1pd,u2pd,u3pd,parametros)
M=parametros(1);
g=parametros(2);

F3d=M*sqrt(u1pd^2+u2pd^2+(u3pd+g)^2);
q5d=asin(M*(u1pd/F3d));
q4d=-asin(M*(u2pd/(F3d*cos(q5d))));
```

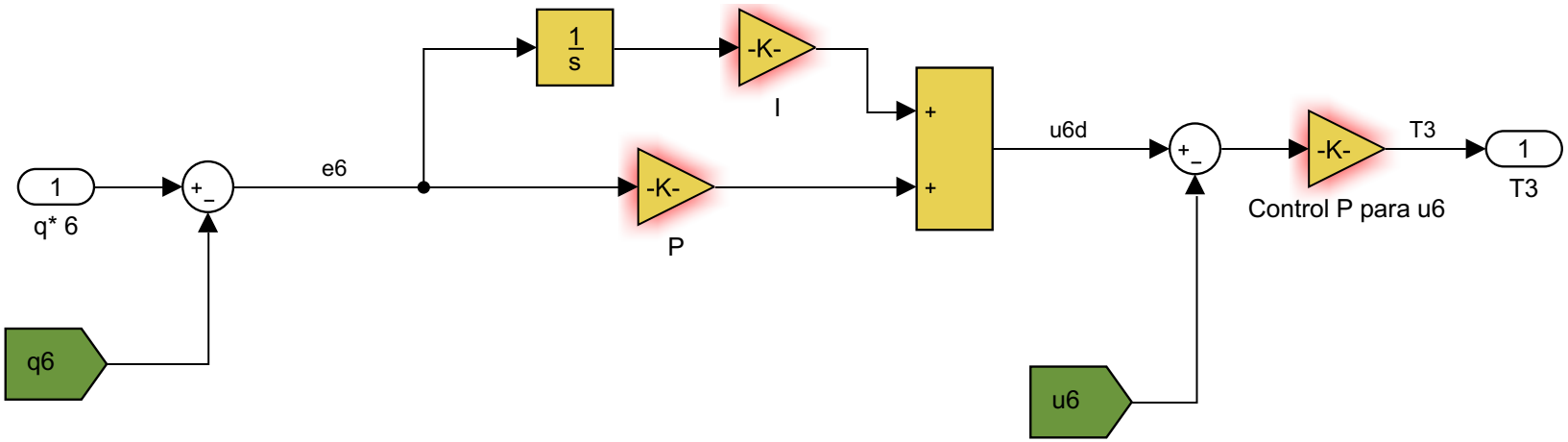


uav_3d/Controlador y sistema/Control traslación/Q inv	7

```
function [u4d,u5d] = Qinv(q4pd,q5pd,q5,q6)

u4d=q4pd*cos(q6)*cos(q5)+q5pd*sin(q6);
u5d=q5pd*cos(q6)-q4pd*sin(q6)*cos(q5);
```

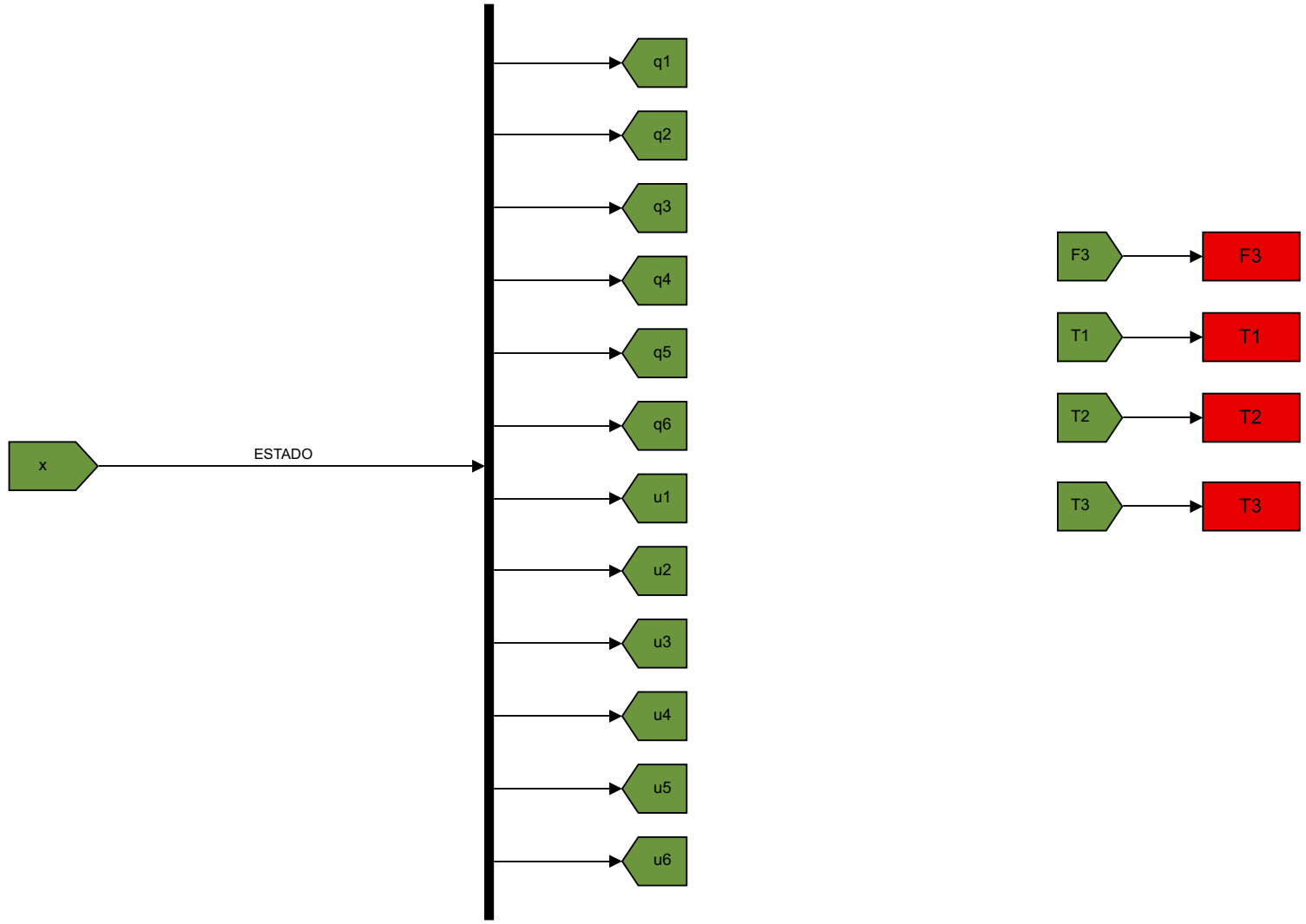


uav_3d/Controlador y sistema/Control traslación/Rtrans	9



uav_3d/Controlador y sistema/Control yaw	10

```
function xp = modelodin(F3,T1,T2,T3,x,parametros)

%ENTRADAS

q1=x(1); q2=x(2); q3=x(3); q4=x(4); q5=x(5); q6=x(6);
u1=x(7); u2=x(8); u3=x(9); u4=x(10); u5=x(11); u6=x(12);

M=parametros(1);
g=parametros(2);
I11=parametros(3);
I22=parametros(4);
I33=parametros(5);

%CONDICIONES

F1=0;   F2=0;

Cfn=[cos(q5)*cos(q6),                           -cos(q5)*sin(q6),                         sin(q5)         ;...
     cos(q4)*sin(q6)+sin(q4)*sin(q5)*cos(q6),   cos(q4)*cos(q6)-sin(q4)*sin(q5)*sin(q6), -sin(q4)*cos(q5);...
     sin(q4)*sin(q6)-cos(q4)*sin(q5)*cos(q6),   sin(q4)*cos(q6)+cos(q4)*sin(q5)*sin(q6), cos(q4)*cos(q5) ];

%ECUACIONES CINEMÁTICAS

q1p=u1;
q2p=u2;
q3p=u3;
q4p=(u4*cos(q6)-u5*sin(q6))/cos(q5);
q5p=u4*sin(q6)+u5*cos(q6);
q6p=u6+tan(q5)*(u4*cos(q6)-u5*sin(q6));

%ECUACIONES DINÁMICAS

% ([u1p u2p u3p]')*M=Cfn*[F1 F2 F3]'+[0 0 -g*M]';
up=(1/M)*(Cfn*[F1,F2,F3]'+[0,0,-g*M]');

u1p=up(1);
u2p=up(2);
u3p=up(3);
u4p=(T1 + (I33-I22)*u5*u6)/I11;
u5p=(T2 - (I33-I11)*u4*u6)/I22;
u6p=(T3 + (I22-I11)*u4*u5)/I33;

%SALIDAS

xp=[q1p,q2p,q3p,q4p,q5p,q6p,u1p,u2p,u3p,u4p,u5p,u6p];
```



uav_3d/ESTADO Y GOTOS	12

ANEXO B

- **Código para generar matrices de D-H:**

```
%construye la matriz Denavit-Hartenberg
%(en funcion de las variables de articulaciones)
function dh=MDH(teta,d,a,alfa)
dh=[cos(teta),  -cos(alfa)*sin(teta),   sin(alfa)*sin(teta),    a*cos(teta);
    sin(teta),  cos(alfa)*cos(teta),    -sin(alfa)*cos(teta),   a*sin(teta);
    0,          sin(alfa),              cos(alfa),              d;
    0,          0,                      0,                      1];
```

- **Código para generar matrices de transformación del sistema:**

```
% transformaciones

clear all
close all

syms q7 q8 l1 l2

A1=MDH(q7,0,l1,0)
A2=MDH(q8,0,l2,0)

T=A1*A2
T=simplify(T)
```

ANEXO C

- **Texto de entrada a autolev con comentarios adicionales:**

```
% File:  UAV_manipulator.al
% quadrotor + manipulador 2 links RR
%--------------------------------------------------------------------
%       Newtonian, bodies, frames, particles and points

NEWTONIAN N     %eje referencia inercial
BODIES    F     %cuerpo con masa (plataforma)
FRAMES    LU,LD     %bases ortonormales ligadas a los dos eslabones del brazo
manipulador
PARTICLES JA,JB     %joints A y B del brazo manipulador

%--------------------------------------------------------------------
%       Variables, constants and specified

VARIABLES  q1',q3',q5',q7',q8'          %velocidades
VARIABLES  FR3,T2,T7,T8             %fuerzas y pares de actuación
CONSTANTS  M,m1,m2,I11,I22,I33,d,l1,l2,g    %constantes

%--------------------------------------------------------------------
%       Motion variables for static/dynamic analysis

MOTIONVARIABLES' u1',u3',u5',u7',u8'        %variables
dinámicas(aceleraciones)

%--------------------------------------------------------------------
%       Mass and inertia properties

MASS     F = M          %le asigna M como masa al cuerpo F
INERTIA  F,I11,I22,I33      %declara las magnitudes inerciales del cuerpo F
MASS     JA = m1        %masa de los eslabones localizada en los joints
MASS     JB = m2

%--------------------------------------------------------------------
%       Geometry relating unit vectors

SIMPROT(N,F,2,q5)   %Matriz de giro N_F creada segun un giro de q5 rad segun
N2> o F2>
SIMPROT(F,LU,2,q7)  %Matriz de firo F_LU creada segun un giro de q7 rad segun
F2> o LU2>
SIMPROT(LU,LD,2,q8) %Matriz de giro LU_LD creada segun un giro de q8 rad
segun LU2> o LD2>

%--------------------------------------------------------------------
%       Angular velocities

W_F_N>   = u5*F2>   %Vectores de velocidad angular (sentido positivo de giro
según los ejes '2')
W_LU_F>  = u7*LU2>
W_LD_LU> = u8*LD2>

%--------------------------------------------------------------------
%       Kinematical differential equations (if any)

q1'=u1      %Ecuaciones cinemáticas
q3'=u3
q5'=u5
q7'=u7
q8'=u8
```

```
%----------------------------------------------------------------------
%       Position vectors

P_NO_FO>   = q1*N1> + q3*N3>    %Vector de posición desde el origen de N
hasta el origen de F
P_FO_LUO>  = -d*F3>     %Vector de posición desde el origen de F hasta el
origen de LU (LU=sistema ejes eslabón 1, situado en parte inferior de
plataforma)
P_LUO_LDO> = l1*LU1>        %Vector de posición desde el origen de LU hasta
el origen de LD (LD=sistema ejes eslabón 2, situado en la segunda
articulación)
P_LDO_JA>  = 0>         %Vector de posición desde el origen de LD hasta la
localización de la masa del eslabón 1 (m1 se coloca en el extremo del eslabón
1)
P_LDO_JB>  = l2*LD1>        %Vector de posición desde el origen de LD hasta
la localización de la masa del eslabón 2 (m2 se coloca en el extremo del
eslabón 2)
%----------------------------------------------------------------------
%       Velocities

V_FO_N> = u1*N1> + u3*N3>   %Vector velocidad de F respecto a N
V2PTS(N,F,FO,LUO)       %Vector V_LU0_N = velocidad de LU0 respecto a N ->
usa la formula conocida de velocidad de 2 puntos fijos de un solido rigido
V2PTS(N,LU,LUO,LDO)     %Vector V_LD0_N = velocidad de LD0 respecto a N
V_JA_N> = V_LDO_N>      %Vector velocidad de JA respecto a N
V2PTS(N,LD,LDO,JB)      Vector V_JB_N = velocidad de JB respecto a N

%----------------------------------------------------------------------
%       Configuration constraints (if any)

%----------------------------------------------------------------------
%       Motion constraints (if any)

%----------------------------------------------------------------------
%       Angular accelerations

ALF_F_N>  = DT(W_F_N>,N)    %(alfa de F respecto a N) es la derivada respecto
al tiempo de (omega de F respecto a N ) }  segun n
ALF_LU_N> = DT(W_LU_N>,N)
ALF_LD_N> = DT(W_LD_N>,N)

%----------------------------------------------------------------------
%       Accelerations

A_FO_N> = DT(V_FO_N>,N)     %aceleracion de F respecto a N
A2PTS(N,F,FO,LUO)       %Aceleracion de LU0 respecto a N    usa la formula
conocida de la aceleracion de 2 puntos fijos a un solido rigido
A2PTS(N,LU,LUO,LDO)
A_JA_N> = A_LDO_N>
A2PTS(N,LD,LDO,JB)

%----------------------------------------------------------------------
%       Forces

GRAVITY(-g*N3>)         %aplica fuerza de gravedad segun N3 a todos los
cuerpos y partículas
FORCE_FO> += FR3*F3>        %Introduce la fuerza de sustentamiento del UAV
(F3) que tiene el sentido vertical según los ejes solidarios al dron ({F})

%----------------------------------------------------------------------
```

```
%       Torques

TORQUE_F> += T2*F2>     %aplica los pares sobre F (para el dron en 2D solo
tenemos T2 según el eje solidario F2> )
TORQUE(F/LU,T7*LU2>)        %añade -T7*LU2> al cuerpo del dron (F) y +T7*LU2
al primer eslabón (par del motor se aplica a los dos con signo contrario)
TORQUE(LU/LD,T8*LD2>)       %idem entre eslabón 1 y eslabón 2, con T8

%----------------------------------------------------------------------
%       Equations of motion

ZERO=FR()+FRSTAR()

KANE()                  %Genera las ecuaciones de movimiento que buscamos

CHECK = NICHECK()

CODE Dynamics() dynamics_coded.all  %Genera el fichero .m con las ecuaciones
dinamicas resueltas segun aceleraciones (implementación del modelo en
simulink)

%----------------------------------------------------------------------
%       Record Autolev responses

Save output_UAV_manipulator_comentado.all

%----------------------------------------------------------------------
```

- **Salida generada por el programa con la dinámica resuelta:**

```
function dinamycs_coded
SolveOrdinaryDifferentialEquations
% File  dinamycs_coded.m  created by Autolev 4.1 on Thu Sep 05 11:31:16 2019


%=============================================================================
function VAR = ReadUserInput
global   d g I22 l1 l2 M m1 m2 FR3 T2 T7 T8;
global   q1 q3 q5 q7 q8 u1 u3 u5 u7 u8 WCHECK1;
global   q1p q3p q5p q7p q8p u1p u3p u5p u7p u8p WCHECK1p;
global   DEGtoRAD RADtoDEG COEF RHS SolutionToAxEqualsB;
global   TINITIAL TFINAL INTEGSTP PRINTINT ABSERR RELERR;

%------------------------------+--------------------------+-----------------
--+-----------------
% Quantity                     | Value                    | Units
| Description
%------------------------------|--------------------------|-----------------
--|-----------------
d                              =  0.0;                    % UNITS
Constant
g                              =  0.0;                    % UNITS
Constant
I22                            =  0.0;                    % UNITS
Constant
l1                             =  0.0;                    % UNITS
Constant
l2                             =  0.0;                    % UNITS
Constant
```

```
M                               =  0.0;                  % UNITS
Constant
m1                              =  0.0;                  % UNITS
Constant
m2                              =  0.0;                  % UNITS
Constant
FR3                             =  0.0;                  % UNITS
Constant
T2                              =  0.0;                  % UNITS
Constant
T7                              =  0.0;                  % UNITS
Constant
T8                              =  0.0;                  % UNITS
Constant

q1                              =  0.0;                  % UNITS
Initial Value
q3                              =  0.0;                  % UNITS
Initial Value
q5                              =  0.0;                  % UNITS
Initial Value
q7                              =  0.0;                  % UNITS
Initial Value
q8                              =  0.0;                  % UNITS
Initial Value
u1                              =  0.0;                  % UNITS
Initial Value
u3                              =  0.0;                  % UNITS
Initial Value
u5                              =  0.0;                  % UNITS
Initial Value
u7                              =  0.0;                  % UNITS
Initial Value
u8                              =  0.0;                  % UNITS
Initial Value
WCHECK1                         =  0.0;                  % UNITS
Initial Value

TINITIAL                        =  0.0;                  % UNITS
Initial Time
TFINAL                          =  1.0;                  % UNITS
Final Time
INTEGSTP                        =  0.1;                  % UNITS
Integration Step
PRINTINT                        =  1;                    % Positive Integer
Print-Integer
ABSERR                          =  1.0E-08;              %
Absolute Error
RELERR                          =  1.0E-07 ;             %
Relative Error
%-------------------------------+--------------------------+-----------------
--+-----------------

% Unit conversions
Pi       = 3.141592653589793;
DEGtoRAD = Pi/180.0;
RADtoDEG = 180.0/Pi;

% Reserve space and initialize matrices
COEF = zeros(5,5);
RHS = zeros(1,5);
```

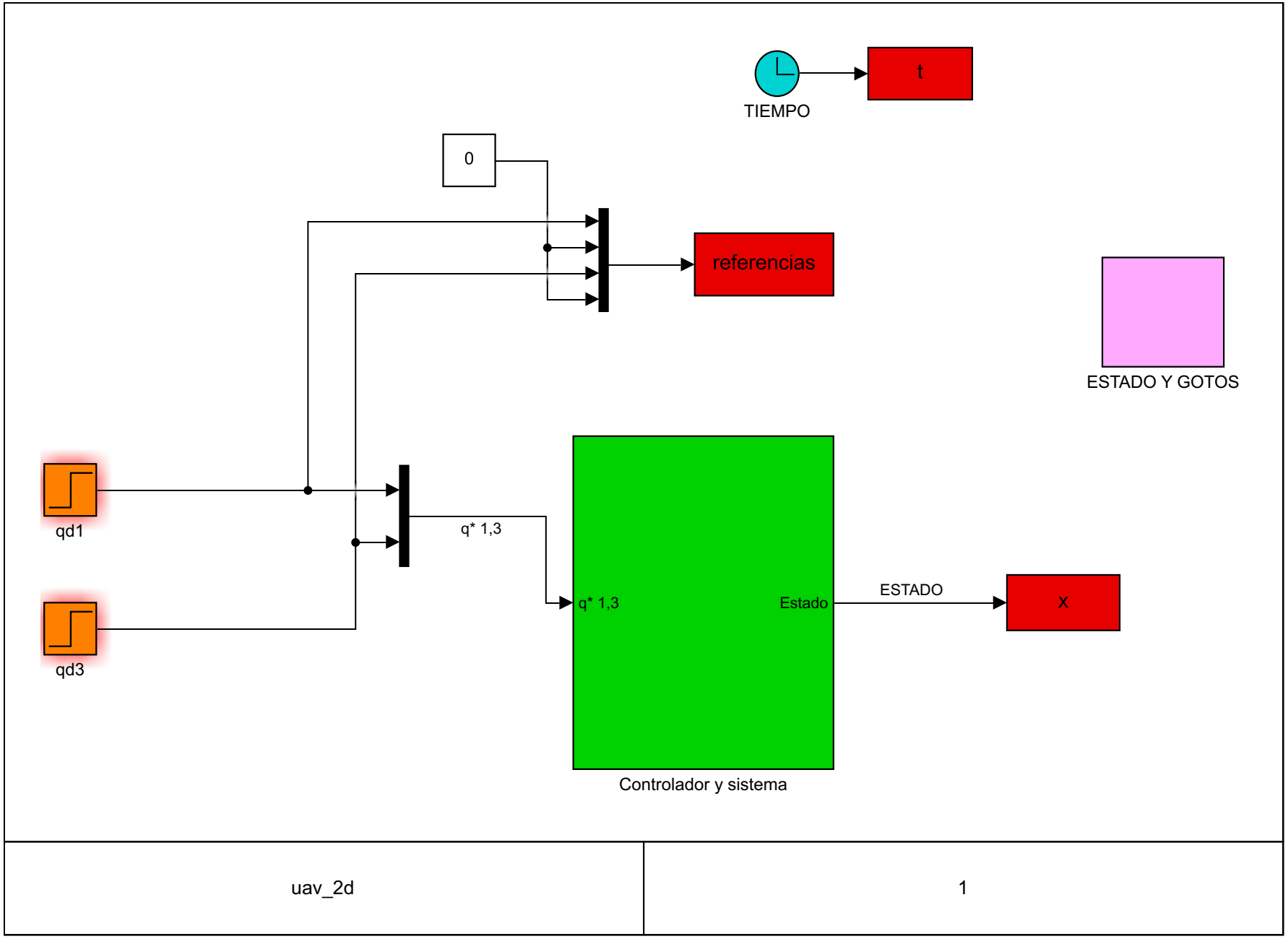
```
% Evaluate constants
% Set the initial values of the states
VAR(1) = q1;
VAR(2) = q3;
VAR(3) = q5;
VAR(4) = q7;
VAR(5) = q8;
VAR(6) = u1;
VAR(7) = u3;
VAR(8) = u5;
VAR(9) = u7;
VAR(10) = u8;
VAR(11) = WCHECK1;



%=============================================================================
% Main driver loop for numerical integration of differential equations
%=============================================================================
function SolveOrdinaryDifferentialEquations
global   d g I22 l1 l2 M m1 m2 FR3 T2 T7 T8;
global   q1 q3 q5 q7 q8 u1 u3 u5 u7 u8 WCHECK1;
global   q1p q3p q5p q7p q8p u1p u3p u5p u7p u8p WCHECK1p;
global   DEGtoRAD RADtoDEG COEF RHS SolutionToAxEqualsB;
global   TINITIAL TFINAL INTEGSTP PRINTINT ABSERR RELERR;

VAR = ReadUserInput;
OdeMatlabOptions = odeset( 'RelTol',RELERR, 'AbsTol',ABSERR,
'MaxStep',INTEGSTP );
T = TINITIAL;
PrintCounter = 0;
mdlDerivatives(T,VAR,0);
while 1,
  if( TFINAL>=TINITIAL & T+0.01*INTEGSTP>=TFINAL ) PrintCounter = -1; end
  if( TFINAL<=TINITIAL & T+0.01*INTEGSTP<=TFINAL ) PrintCounter = -1; end
  if( PrintCounter <= 0.01 ),
     mdlOutputs(T,VAR,0);
     if( PrintCounter == -1 ) break; end
     PrintCounter = PRINTINT;
  end
  [TimeOdeArray,VarOdeArray] = ode45( @mdlDerivatives, [T T+INTEGSTP], VAR,
OdeMatlabOptions, 0 );
  TimeAtEndOfArray = TimeOdeArray( length(TimeOdeArray) );
  if( abs(TimeAtEndOfArray - (T+INTEGSTP) ) >= abs(0.001*INTEGSTP) )
warning('numerical integration failed'); break;  end
  T = TimeAtEndOfArray;
  VAR = VarOdeArray( length(TimeOdeArray), : );
  PrintCounter = PrintCounter - 1;
end
mdlTerminate(T,VAR,0);



%=============================================================================
% mdlDerivatives: Calculates and returns the derivatives of the continuous
states
%=============================================================================
function sys = mdlDerivatives(T,VAR,u)
global   d g I22 l1 l2 M m1 m2 FR3 T2 T7 T8;
global   q1 q3 q5 q7 q8 u1 u3 u5 u7 u8 WCHECK1;
global   q1p q3p q5p q7p q8p u1p u3p u5p u7p u8p WCHECK1p;
```

```
global   DEGtoRAD RADtoDEG COEF RHS SolutionToAxEqualsB;
global   TINITIAL TFINAL INTEGSTP PRINTINT ABSERR RELERR;

% Update variables after integration step
q1 = VAR(1);
q3 = VAR(2);
q5 = VAR(3);
q7 = VAR(4);
q8 = VAR(5);
u1 = VAR(6);
u3 = VAR(7);
u5 = VAR(8);
u7 = VAR(9);
u8 = VAR(10);
WCHECK1 = VAR(11);
q1p = u1;
q3p = u3;
q5p = u5;
q7p = u7;
q8p = u8;
WCHECK1p = (g*M+g*m1+g*m2-FR3*cos(q5))*u3 - FR3*sin(q5)*u1 -
(T8+g*l2*m2*cos(q5+q7+q8))*u8 -
(T7+g*l1*m1*cos(q5+q7)+g*m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8)))*u7 - (T2-
g*m1*(d*sin(q5)-l1*cos(q5+q7))-g*m2*(d*sin(q5)-l1*cos(q5+q7)-
l2*cos(q5+q7+q8)))*u5;

COEF(1,1) = -M - m1 - m2;
COEF(1,2) = 0;
COEF(1,3) = m1*(d*cos(q5)+l1*sin(q5+q7)) +
m2*(d*cos(q5)+l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(1,4) = l1*m1*sin(q5+q7) + m2*(l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(1,5) = l2*m2*sin(q5+q7+q8);
COEF(2,1) = 0;
COEF(2,2) = -M - m1 - m2;
COEF(2,3) = -m1*(d*sin(q5)-l1*cos(q5+q7)) - m2*(d*sin(q5)-l1*cos(q5+q7)-
l2*cos(q5+q7+q8));
COEF(2,4) = l1*m1*cos(q5+q7) + m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8));
COEF(2,5) = l2*m2*cos(q5+q7+q8);
COEF(3,1) = m1*(d*cos(q5)+l1*sin(q5+q7)) +
m2*(d*cos(q5)+l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(3,2) = -m1*(d*sin(q5)-l1*cos(q5+q7)) - m2*(d*sin(q5)-l1*cos(q5+q7)-
l2*cos(q5+q7+q8));
COEF(3,3) = -I22 - m1*(d^2+l1^2+2*d*l1*sin(q7)) -
m2*(d^2+l1^2+l2^2+2*d*l1*sin(q7)+2*l1*l2*cos(q8)+2*d*l2*sin(q7+q8));
COEF(3,4) = -l1*m1*(l1+d*sin(q7)) -
m2*(l1^2+l2^2+d*l1*sin(q7)+2*l1*l2*cos(q8)+d*l2*sin(q7+q8));
COEF(3,5) = -l2*m2*(l2+l1*cos(q8)+d*sin(q7+q8));
COEF(4,1) = l1*m1*sin(q5+q7) + m2*(l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(4,2) = l1*m1*cos(q5+q7) + m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8));
COEF(4,3) = -l1*m1*(l1+d*sin(q7)) -
m2*(l1^2+l2^2+d*l1*sin(q7)+2*l1*l2*cos(q8)+d*l2*sin(q7+q8));
COEF(4,4) = -m1*l1^2 - m2*(l1^2+l2^2+2*l1*l2*cos(q8));
COEF(4,5) = -l2*m2*(l2+l1*cos(q8));
COEF(5,1) = l2*m2*sin(q5+q7+q8);
COEF(5,2) = l2*m2*cos(q5+q7+q8);
COEF(5,3) = -l2*m2*(l2+l1*cos(q8)+d*sin(q7+q8));
COEF(5,4) = -l2*m2*(l2+l1*cos(q8));
COEF(5,5) = -m2*l2^2;
RHS(1) = m1*(d*sin(q5)*u5^2-l1*cos(q5+q7)*(u5+u7)^2) + m2*(d*sin(q5)*u5^2-
l1*cos(q5+q7)*(u5+u7)^2-l2*cos(q5+q7+q8)*(u5+u7+u8)^2) - FR3*sin(q5);
```

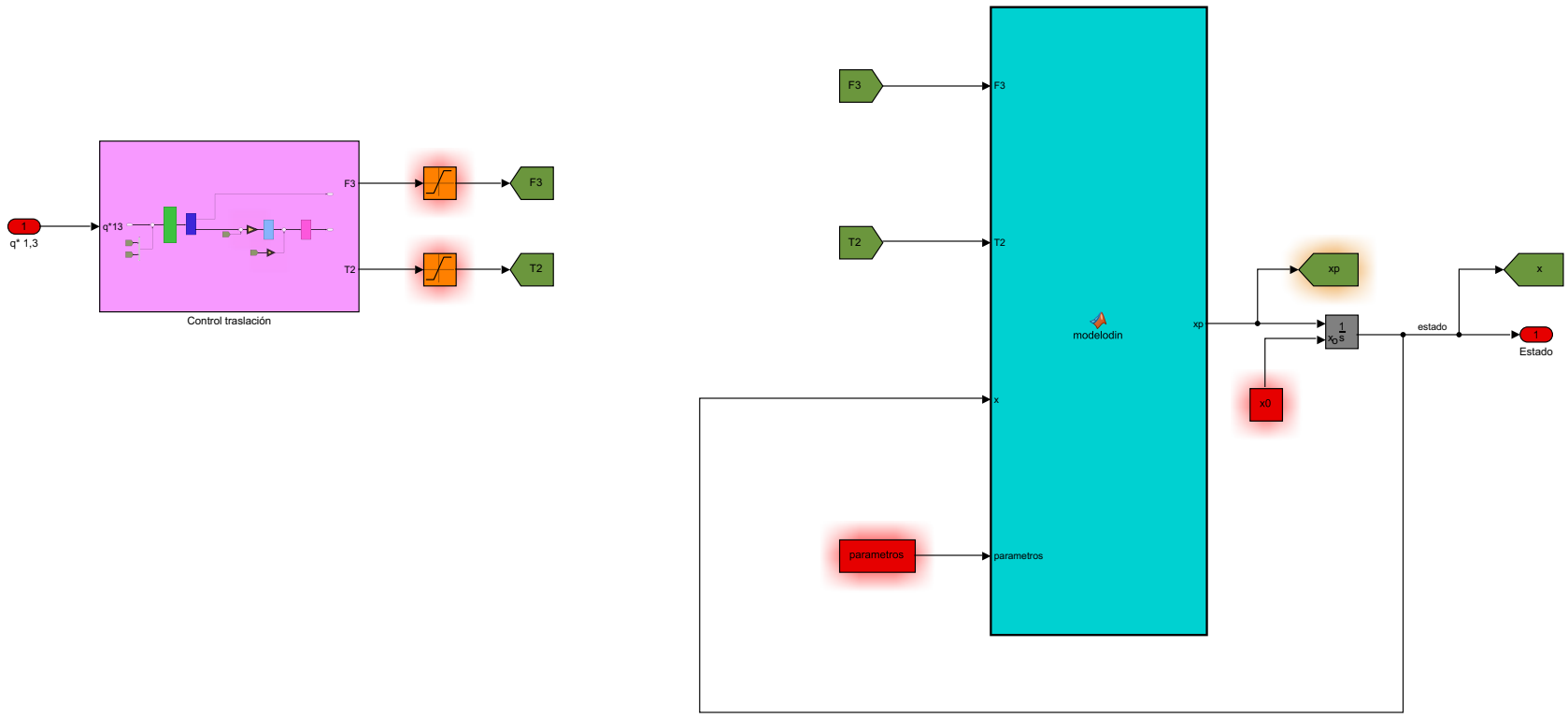
```
RHS(2) = g*M + g*m1 + g*m2 + m1*(d*cos(q5)*u5^2+l1*sin(q5+q7)*(u5+u7)^2) +
m2*(d*cos(q5)*u5^2+l1*sin(q5+q7)*(u5+u7)^2+l2*sin(q5+q7+q8)*(u5+u7+u8)^2) -
FR3*cos(q5);
RHS(3) = g*m1*(d*sin(q5)-l1*cos(q5+q7)) + g*m2*(d*sin(q5)-l1*cos(q5+q7)-
l2*cos(q5+q7+q8)) - T2 - d*l1*m1*cos(q7)*(u5^2-(u5+u7)^2) -
m2*(d*l1*cos(q7)*u5^2+d*l2*cos(q7+q8)*u5^2+l1*l2*sin(q8)*(u5+u7+u8)^2-
d*l1*cos(q7)*(u5+u7)^2-l1*l2*sin(q8)*(u5+u7)^2-d*l2*cos(q7+q8)*(u5+u7+u8)^2);
RHS(4) = m2*(l1*l2*sin(q8)*(u5+u7)^2-d*l1*cos(q7)*u5^2-d*l2*cos(q7+q8)*u5^2-
l1*l2*sin(q8)*(u5+u7+u8)^2) - T7 - g*l1*m1*cos(q5+q7) -
g*m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8)) - d*l1*m1*cos(q7)*u5^2;
RHS(5) = -T8 - g*l2*m2*cos(q5+q7+q8) - l2*m2*(d*cos(q7+q8)*u5^2-
l1*sin(q8)*(u5+u7)^2);
SolutionToAxEqualsB = COEF\RHS';

% Update variables after uncoupling equations
u1p = SolutionToAxEqualsB(1);
u3p = SolutionToAxEqualsB(2);
u5p = SolutionToAxEqualsB(3);
u7p = SolutionToAxEqualsB(4);
u8p = SolutionToAxEqualsB(5);

% Update derivative array prior to integration step
VARp(1) = q1p;
VARp(2) = q3p;
VARp(3) = q5p;
VARp(4) = q7p;
VARp(5) = q8p;
VARp(6) = u1p;
VARp(7) = u3p;
VARp(8) = u5p;
VARp(9) = u7p;
VARp(10) = u8p;
VARp(11) = WCHECK1p;

sys = VARp';



%=============================================================================
% mdlOutputs: Calculates and return the outputs
%=============================================================================
function Output = mdlOutputs(T,VAR,u)
global   d g I22 l1 l2 M m1 m2 FR3 T2 T7 T8;
global   q1 q3 q5 q7 q8 u1 u3 u5 u7 u8 WCHECK1;
global   q1p q3p q5p q7p q8p u1p u3p u5p u7p u8p WCHECK1p;
global   DEGtoRAD RADtoDEG COEF RHS SolutionToAxEqualsB;
global   TINITIAL TFINAL INTEGSTP PRINTINT ABSERR RELERR;

% Evaluate output quantities



%=============================================================================
function WriteOutput( fileIdentifier, Output )
numberOfOutputQuantities = length( Output );
if numberOfOutputQuantities > 0,
  for i=1:numberOfOutputQuantities,
    fprintf( fileIdentifier, ' %- 14.6E', Output(i) );
  end
  fprintf( fileIdentifier, '\n' );
end
```

```
%=============================================================================
% mdlTerminate: Perform end of simulation tasks and set sys=[]
%=============================================================================
function sys = mdlTerminate(T,VAR,u)
sys = [];



%=============================================================================
% Sfunction: System/Simulink function from standard template
%=============================================================================
function [sys,x0,str,ts] = Sfunction(t,x,u,flag)
switch flag,
  case 0,  [sys,x0,str,ts] = mdlInitializeSizes;    % Initialization of sys,
initial state x0, state ordering string str, and sample times ts
  case 1,  sys = mdlDerivatives(t,x,u);             % Calculate the
derivatives of continuous states and store them in sys
  case 2,  sys = mdlUpdate(t,x,u);                  % Update discrete states
x(n+1) in sys
  case 3,  sys = mdlOutputs(t,x,u);                 % Calculate outputs in
sys
  case 4,  sys = mdlGetTimeOfNextVarHit(t,x,u);     % Return next sample time
for variable-step in sys
  case 9,  sys = mdlTerminate(t,x,u);               % Perform end of
simulation tasks and set sys=[]
  otherwise error(['Unhandled flag = ',num2str(flag)]);
end



%=============================================================================
% mdlInitializeSizes: Return the sizes, initial state VAR, and sample times
ts
%=============================================================================
function [sys,VAR,stateOrderingStrings,timeSampling] = mdlInitializeSizes
sizes = simsizes;             % Call simsizes to create a sizes structure
sizes.NumContStates  = 11;    % sys(1) is the number of continuous states
sizes.NumDiscStates  = 0;     % sys(2) is the number of discrete states
sizes.NumOutputs     = 0;     % sys(3) is the number of outputs
sizes.NumInputs      = 0;     % sys(4) is the number of inputs
sizes.DirFeedthrough = 1;     % sys(6) is 1, and allows for the output to be
a function of the input
sizes.NumSampleTimes = 1;     % sys(7) is the number of samples times (the
number of rows in ts)
sys = simsizes(sizes);        % Convert it to a sizes array
stateOrderingStrings = [];
timeSampling         = [0 0]; % m-by-2 matrix containing the sample times
VAR = ReadUserInput
```

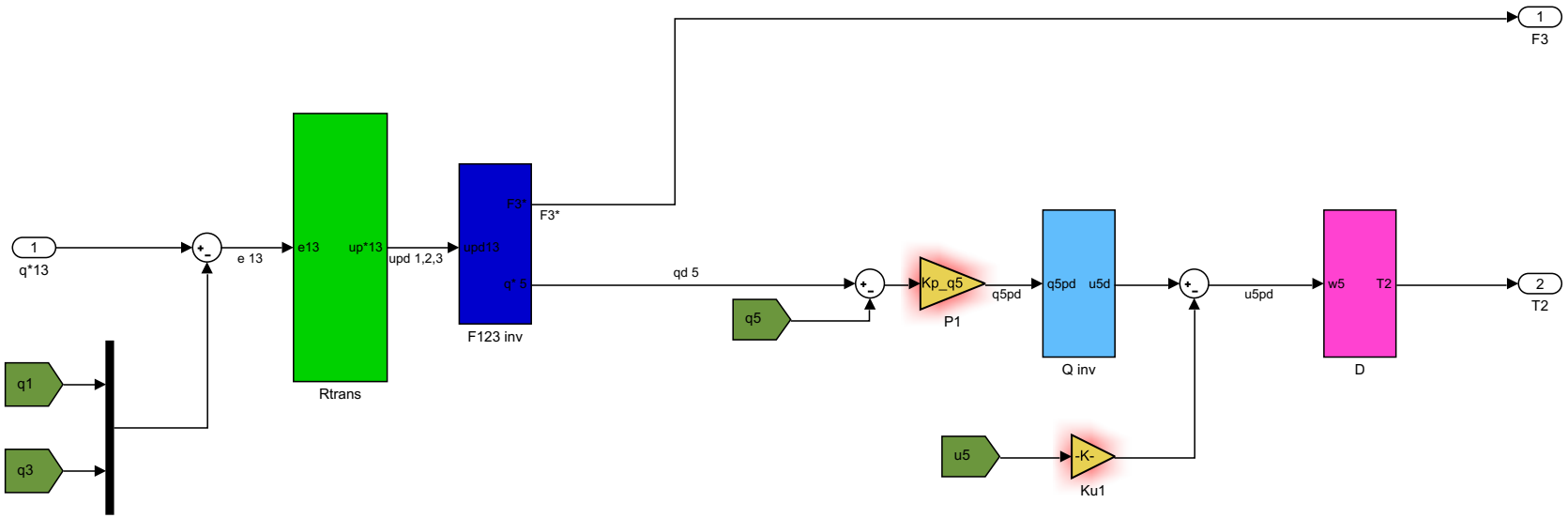
ANEXO D


t
TIEMPO
0
referencias
ESTADO Y GOTOS
qd1
qd3
q* 1,3
q* 1,3
Estado
ESTADO
X
Controlador y sistema

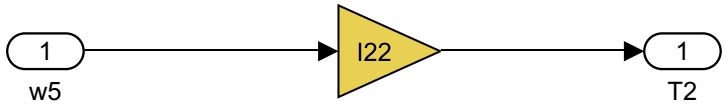
uav_2d	1



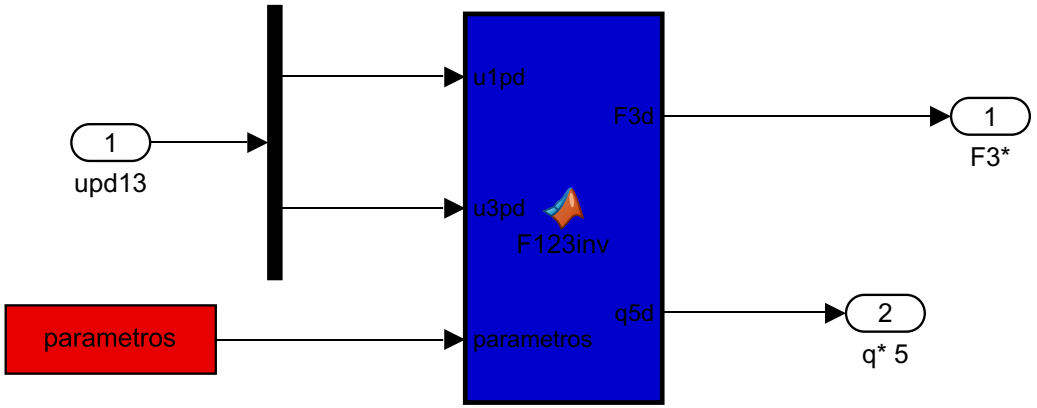
uav_2d/Controlador y sistema	2



uav_2d/Controlador y sistema/Control traslación	3

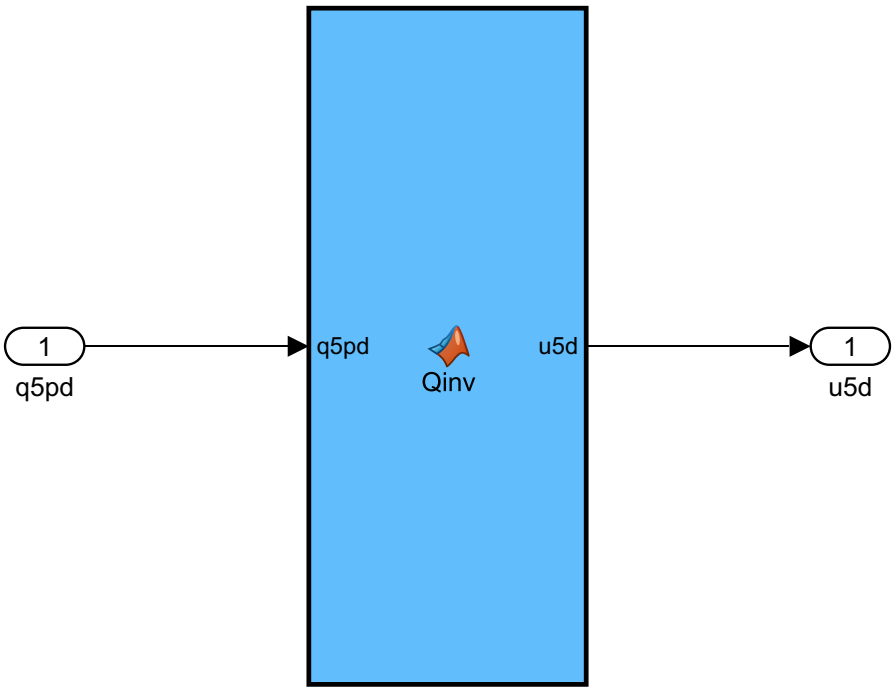


uav_2d/Controlador y sistema/Control traslación/D	4



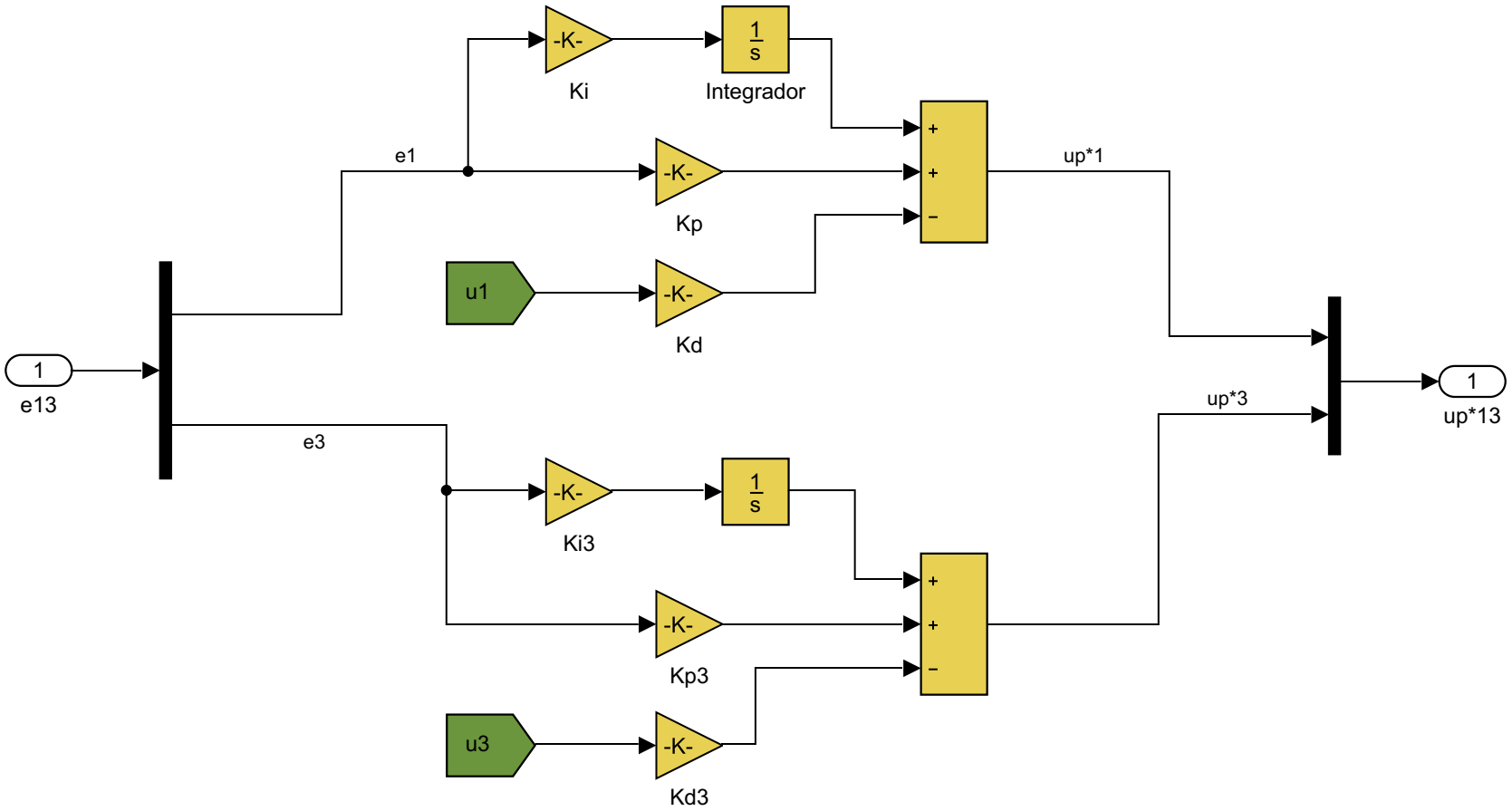
uav_2d/Controlador y sistema/Control traslación/F123 inv	5

```
function [F3d,q5d] = F123inv(u1pd,u3pd,parametros)
M=parametros(1);
g=parametros(2);

F3d=M*sqrt(u1pd^2+(u3pd+g)^2);
q5d=asin(M*(u1pd/F3d));
```



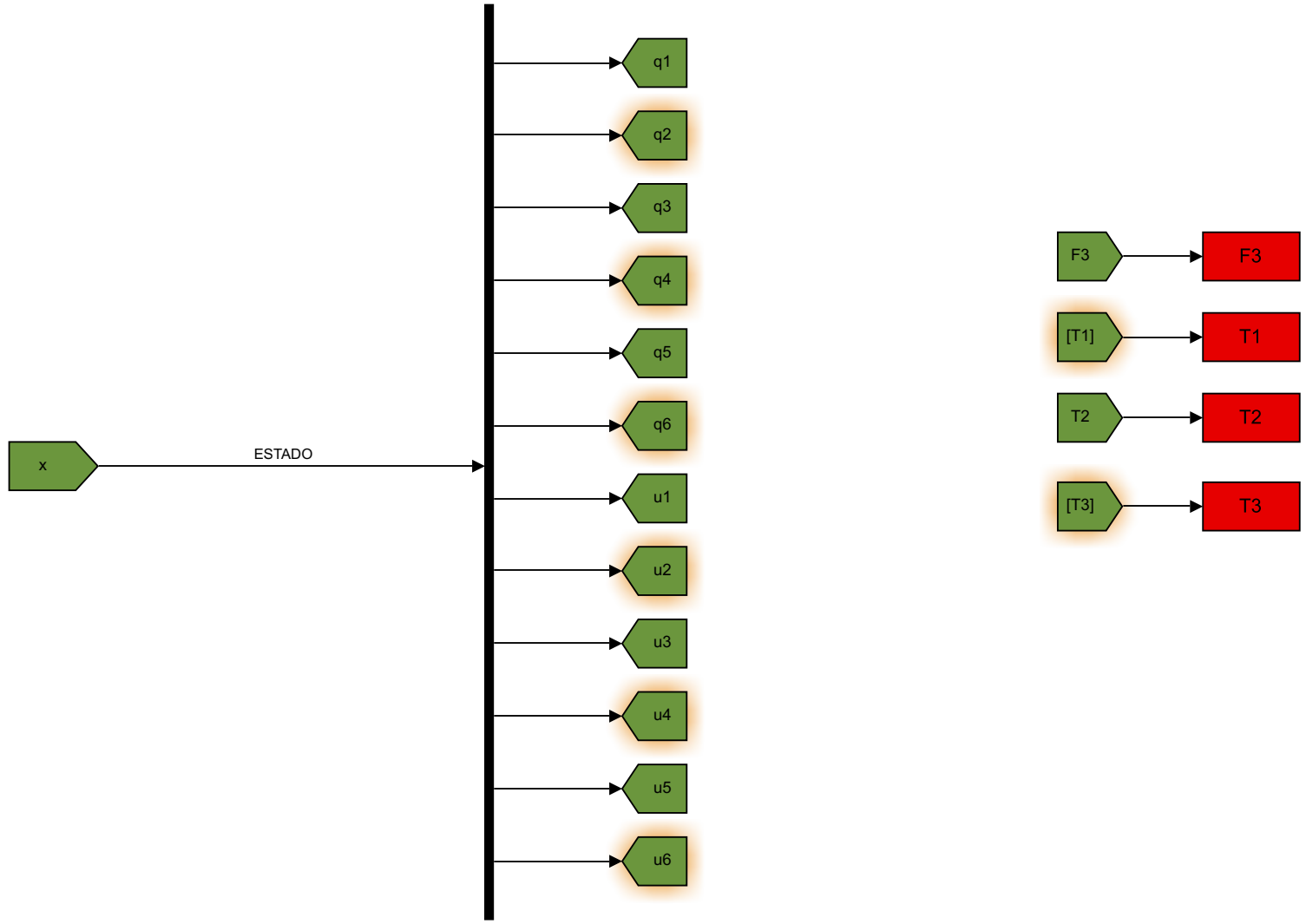
uav_2d/Controlador y sistema/Control traslación/Q inv	7

```
function u5d  = Qinv(q5pd)

u5d=q5pd;
```



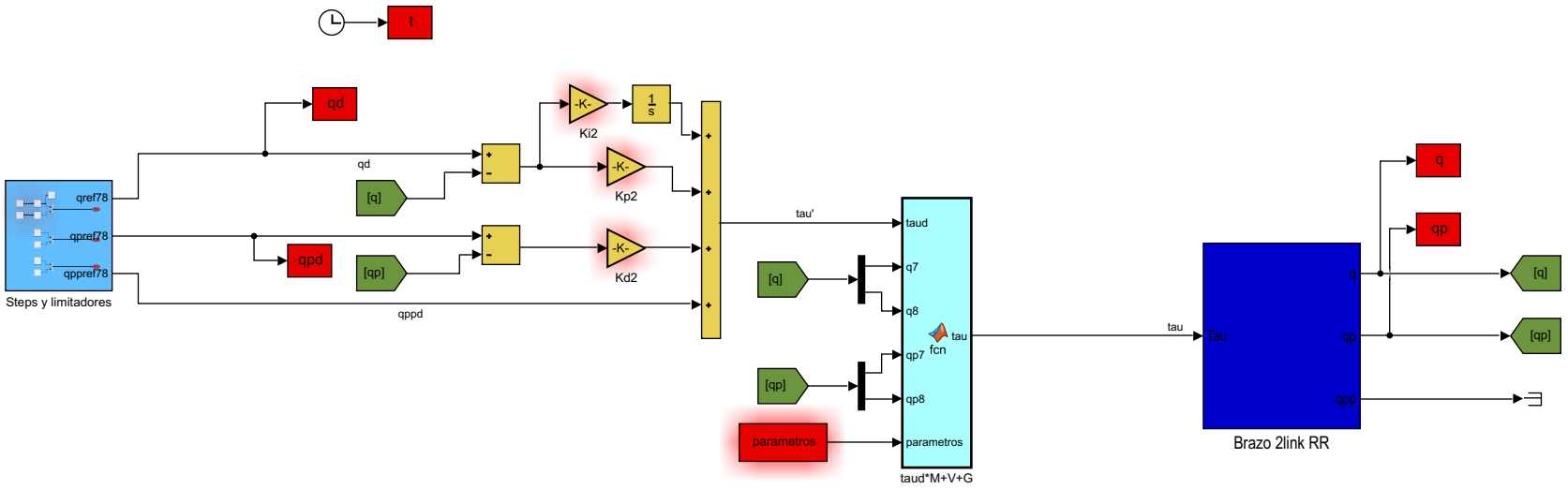
uav_2d/Controlador y sistema/Control traslación/Rtrans	9

```
function xp = modelodin(F3,T2,x,parametros)

%ENTRADAS

q1=x(1); q2=x(2); q3=x(3); q4=x(4); q5=x(5); q6=x(6);
u1=x(7); u2=x(8); u3=x(9); u4=x(10); u5=x(11); u6=x(12);

M=parametros(1);
g=parametros(2);
I11=parametros(3);
I22=parametros(4);
I33=parametros(5);

%CONDICIONES

F1=0;   F2=0;

Cfn=[cos(q5)*cos(q6),                              -cos(q5)*sin(q6),                           sin(q5)         ;...
     cos(q4)*sin(q6)+sin(q4)*sin(q5)*cos(q6),    cos(q4)*cos(q6)-sin(q4)*sin(q5)*sin(q6), -sin(q4)*cos(q5);...
     sin(q4)*sin(q6)-cos(q4)*sin(q5)*cos(q6),    sin(q4)*cos(q6)+cos(q4)*sin(q5)*sin(q6), cos(q4)*cos(q5) ];

%ECUACIONES CINEMÁTICAS

q1p=u1;
q3p=u3;
q5p=u5;


%ECUACIONES DINÁMICAS

% ([u1p u2p u3p]')*M=Cfn*[F1 F2 F3]'+[0 0 -g*M]';
up=(1/M)*(Cfn*[F1,F2,F3]'+[0,0,-g*M]');

u1p=up(1);
u2p=up(2);
u3p=up(3);
u4p=0;
u5p=T2/I22;
u6p=0;

%SALIDAS
q2p=0;q4p=0;q6p=0;
xp=[q1p,q2p,q3p,q4p,q5p,q6p,u1p,u2p,u3p,u4p,u5p,u6p];
```

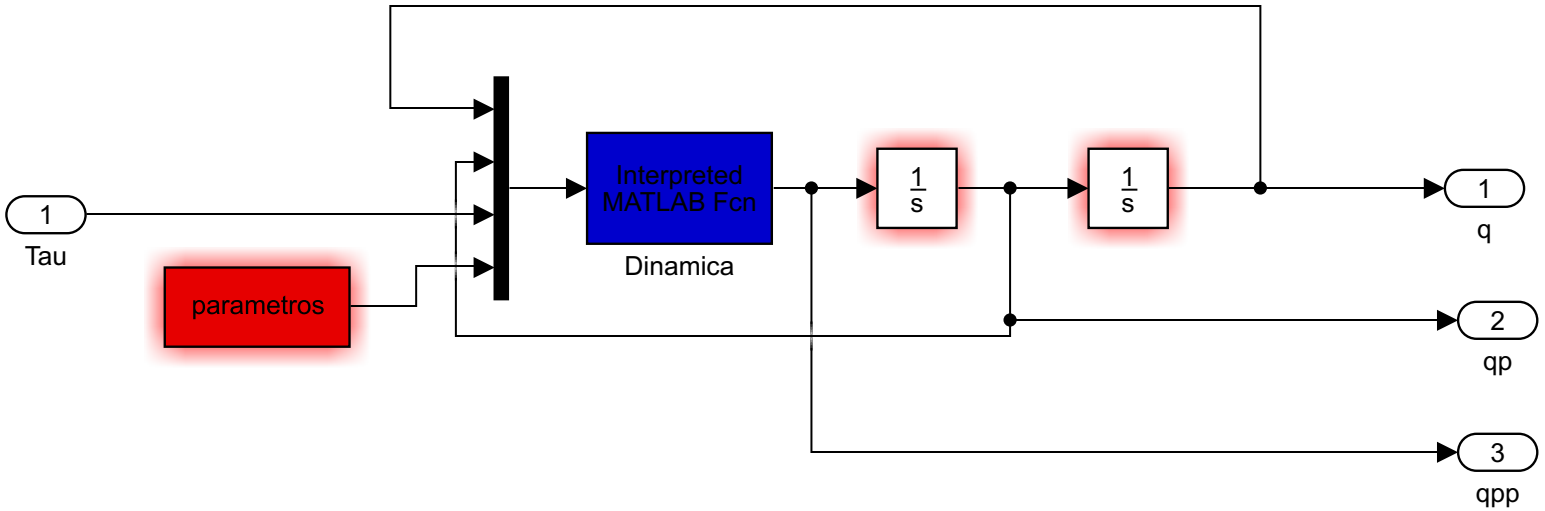


uav_2d/ESTADO Y GOTOS	11

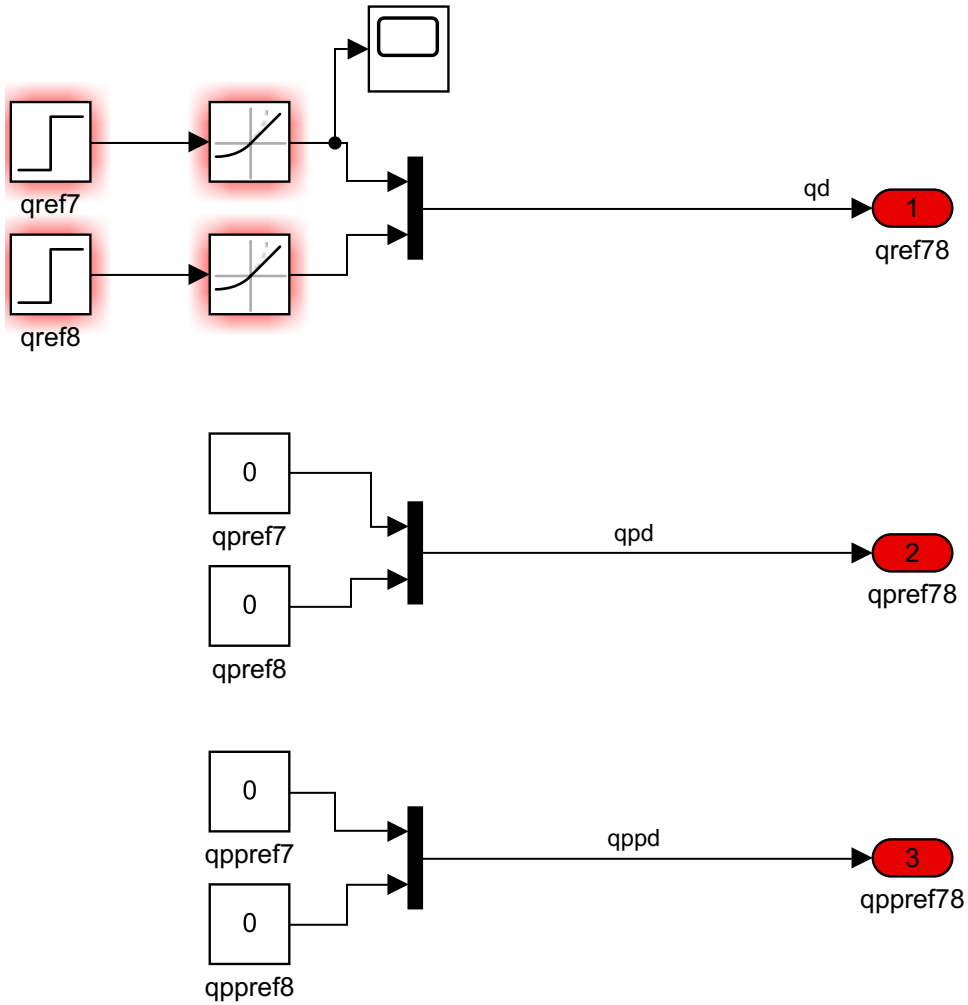
ANEXO E


t
qd
qd
-K-
Ki2
1
s
-K-
Kp2
[q]
qref78
qpref78
qppref78
Steps y limitadores
qpd
[qp]
-K-
Kd2
qppd
tau'
taud
q7
q8
qp7
qp8
parametros
fcn
tau
[q]
[qp]
parametros
taud*M+V+G
tau
Tau
q
qp
qpp
Brazo 2link RR
q
qp
[q]
[qp]

solo_brazo	1



solo_brazo/Brazo 2link RR	2



solo_brazo/Steps y limitadores	3

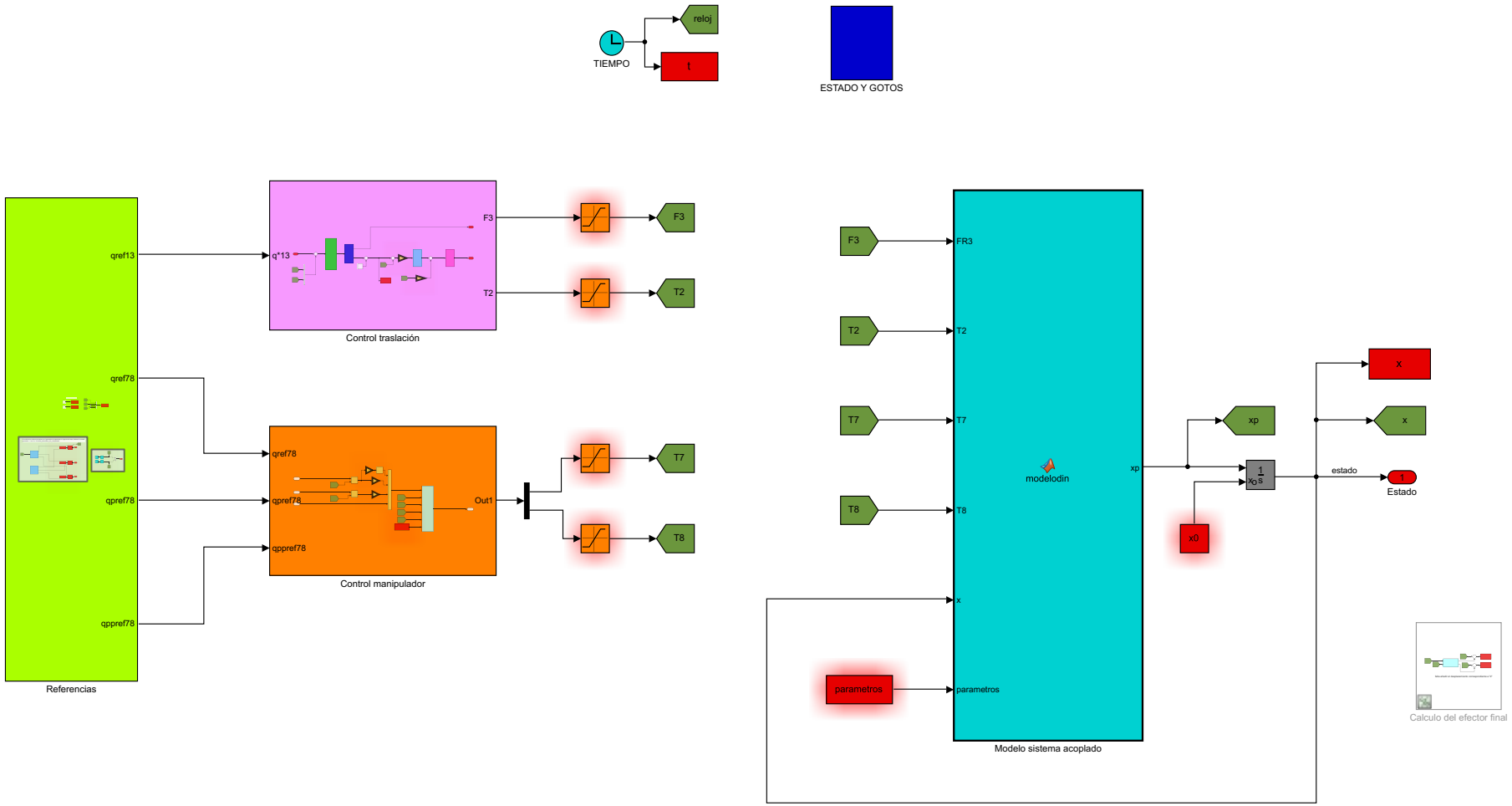
```
function tau = fcn(taud,q7,q8,qp7,qp8,parametros)
tau=[0;0];

%% Parámetros

g=parametros(1);
l1=parametros(2);
l2=parametros(3);
m1=parametros(4);
m2=parametros(5);


%% Cálculo de matrices y vectores
M=[l2^2*m2+2*l1*l2*m2*cos(q8)+l1^2*(m1+m2) , l2^2*m2+l1*l2*m2*cos(q8);...
   l2^2*m2+l1*l2*m2*cos(q8) , l2^2*m2];

V=[-m2*l1*l2*sin(q8)*qp8^2-2*m2*l1*l2*sin(q8)*qp7*qp8;...
   m2*l1*l2*sin(q8)*qp7^2];

G=(-1).*[m2*l2*g*cos(q7+q8)+(m1+m2)*l1*g*cos(q7);...
         m2*l2*g*cos(q7+q8)];


%% Salida
tau=M*taud + V + G;
% tau=taud;   %Usado en muchos controles industrialessegun J.Craig
```

ANEXO F

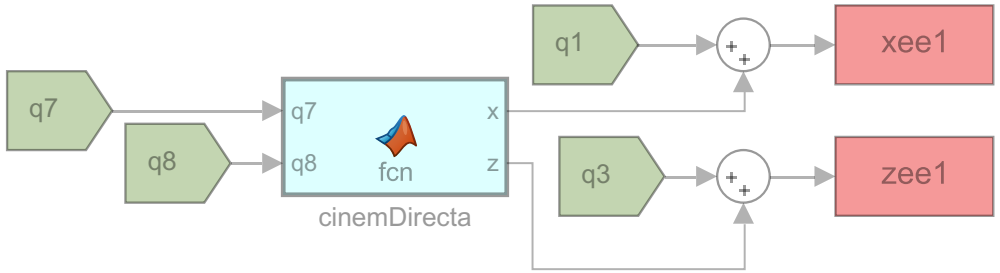
- **El código en el que se declaran todos los parámetros de simulación:**

```
%PARÁMETROS DEL MODELO DE SIMULINK
clear all
%Nota: Todo en unidades del sistema internacional.

%% SIMULACIÓN_____________________________________
t_sim=60;

%Condiciones iniciales
q10=0;
q30=0;
q50=0;
q70=pi/2;       %Se comienza con el brazo hacia abajo
q80=0;
u10=0;
u30=0;
u50=0;
u70=0;
u80=0;
x0=[q10,q30,q50,q70,q80,u10,u30,u50,u70,u80];

%Las referencias están puestas como escalones con t_qrefi,
%qi0 y qrefi.
t_qref1=10;
t_qref3=20;
t_qref7=30;
t_qref8=40;

qref1=1;
qref3=2;
qref7=pi/2 +0.5236; %90+30 °
qref8=pi/3;

%% CONTROLADORES traslacion_____________________________
p1 = -4;   p2 = -4;   p3 = -4;   p4 = -4;   p5 = -4;

%Rori
Kp_q5= p4*p5 + (p4+p5)*p3 + (p3+p4+p5)*p2 + (p2+p3+p4+p5)*p1;

Kp_u5= -(p1+p2+p3+p4+p5);

%Rtrans
Kp_q1=(p2*p3*p4*p5 + p1*(p3*p4*p5 + p2*(p3*(p4+p5) + p4*p5)))/Kp_q5;
% Kp_q1=0.01;
Ki_q1= -p1*p2*p3*p4*p5/Kp_q5;
Kd_q1=-(p3*p4*p5 + p2*(p3*(p4+p5) + p4*p5) + p1*(p4*p5 + p3*(p4+p5) +...
          (p3+p4+p5)*p2))/Kp_q5;

Kp_q3 =   p1*(p2+p3)+p2*p3;
Ki_q3 = - p1*p2*p3;
Kd_q3 = -(p1+p2+p3);

%% DINÁMICA_____________________________________________
I11 = 0.082;
I22 = 0.082;
I33 = 0.149;
M=4;
g=9.81;
```

```
%% SATURACIÓN DE LA ACTUACIÓN_____________________
F3_max=200;
F3_min=0;
T2_max=10;
T2_min=-10;
T7_max=10;
T7_min=-10;
T8_max=10;
T8_min=-10;

%% PARAMETROS DEL BRAZO MANIPULADOR
  L1=0.3;
  L2=0.3;

  m1=0.15;
  m2=0.15;

  d= 0.07;

  i11=m1*L1^2;
  i22=m2*L2^2;

%Parámetros de los motores
  R1=1; %reductoras
  R2=1;

  Jm1=0; %Jm y Bm son parámetros de inercia y rozamiento de
  Jm2=0; %los servomotores que no hemos considerado en nuestro
  Bm1=0; %modelo del manipulador
  Bm2=0;

%_____________---GENERADOR DE TRAYECTORIAS DEL BRAZO DE ROBOTICA 3GITI
generador=0;%1 -> trayectorias por generador
            %0 -> trayectorias con steps y limitadores
t_ini=[t_qref7;t_qref8];
t_fin=t_ini+10;
q_ini=[q70;q80];
q_fin=[qref7;qref8];
qp_ini=[q70;q80];
qp_fin=[u70;u80];


%% BRAZO MANIPULADOR
  Kp_q7=5.108;
  Kd_q7=2.333;
  Ki_q7=2;%0.667;

  Kp_q8=2.372;
  Kd_q8=1.225;
  Ki_q8=0.775;


%% FIN_____________________________________________
parametros=[M,g,I11,I22,I33,L1,L2,m1,m2,d,i11,i22,R1,R2,Jm1,Jm2,Bm1,Bm2];
```

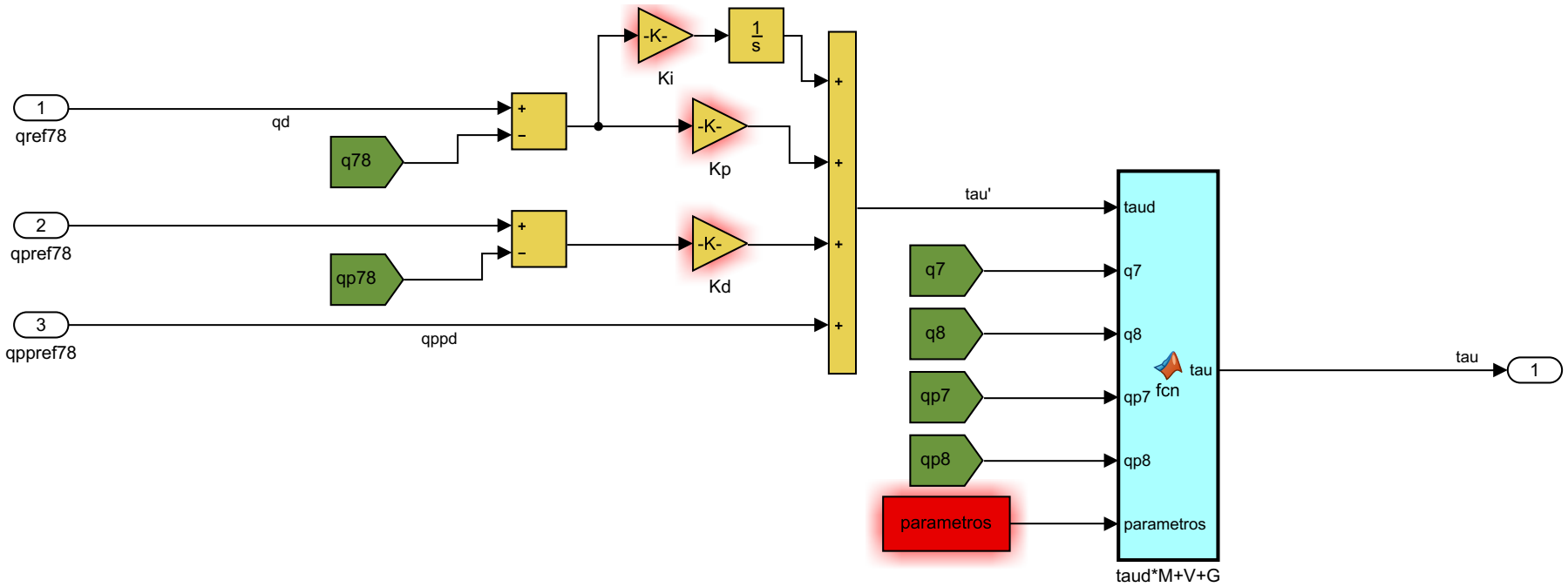


uav_con_manipulador_2d	1



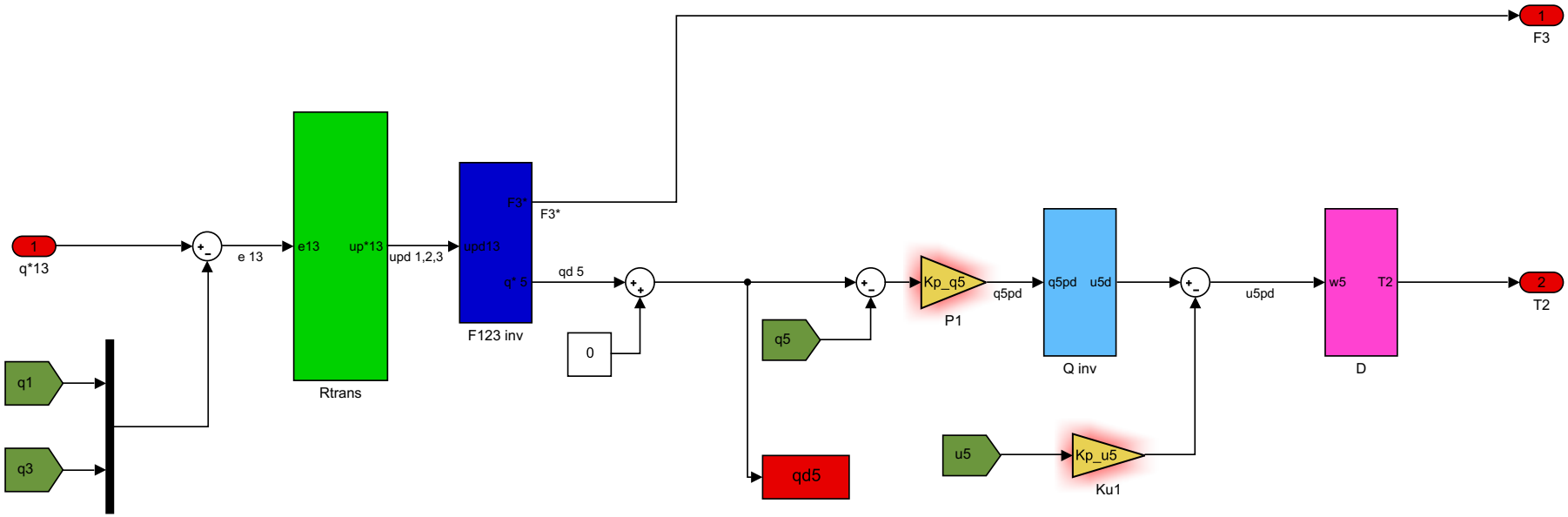
falta añadir el desplazamiento correspondiente a "d"

```
function [x,z]= fcn(q7,q8)
%cinematica directa del brazo
l1=0.64;l2=0.5;

x=l2*cos(q7+q8)+l1*cos(q7);
z=l2*sin(q7+q8)+l1*sin(q7);
```

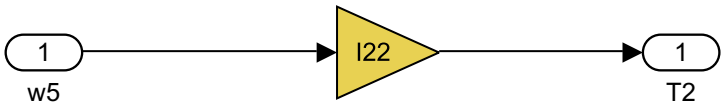


uav_con_manipulador_2d/Control manipulador	4

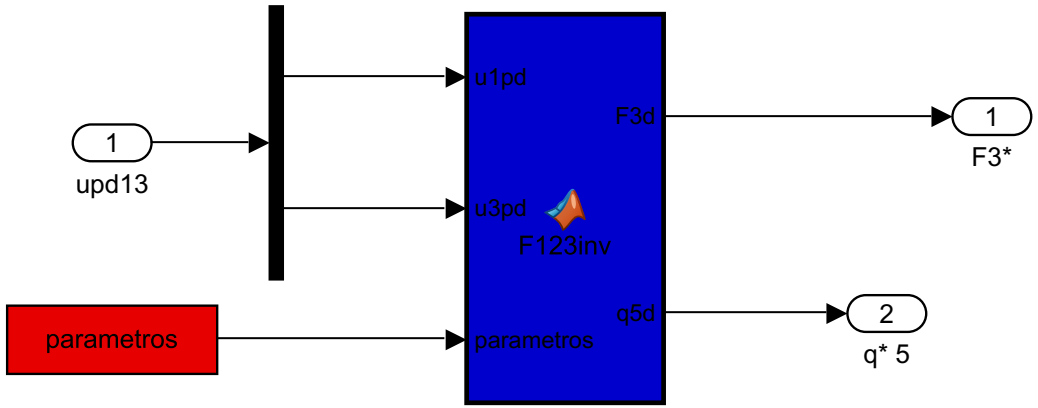
```
function tau = fcn(taud,q7,q8,qp7,qp8,parametros)
tau=[0;0];

%% Parámetros
%             1,2,3,,,,4,,5,,,6,,7,,8,,9,,10,11,12,,13,14,15,,16,,17,,18
% parametros=[M,g,I11,I22,I33,L1,L2,m1,m2,d,i11,i22,R1,R2,Jm1,Jm2,Bm1,Bm2];
g=parametros(2);
l1=parametros(6);
l2=parametros(7);
m1=parametros(8);
m2=parametros(9);
i11=parametros(11);
i22=parametros(12);
R1=parametros(13);
R2=parametros(14);
Jm1=parametros(15);
Jm2=parametros(16);
Bm1=parametros(17);
Bm2=parametros(18);

%% Cálculo de matrices y vectores
M=[l2^2*m2+2*l1*l2*m2*cos(q8)+l1^2*(m1+m2) , l2^2*m2+l1*l2*m2*cos(q8);...
   l2^2*m2+l1*l2*m2*cos(q8) , l2^2*m2];

V=[-m2*l1*l2*sin(q8)*qp8^2-2*m2*l1*l2*sin(q8)*qp7*qp8;...
   m2*l1*l2*sin(q8)*qp7^2];

G=(-1).*[m2*l2*g*cos(q7+q8)+(m1+m2)*l1*g*cos(q7);...
         m2*l2*g*cos(q7+q8)];


%% Salida
tau=M*taud + V + G;
% tau=taud;   %Usado en muchos controles industrialessegun J.Craig
```



uav_con_manipulador_2d/Control traslación	6

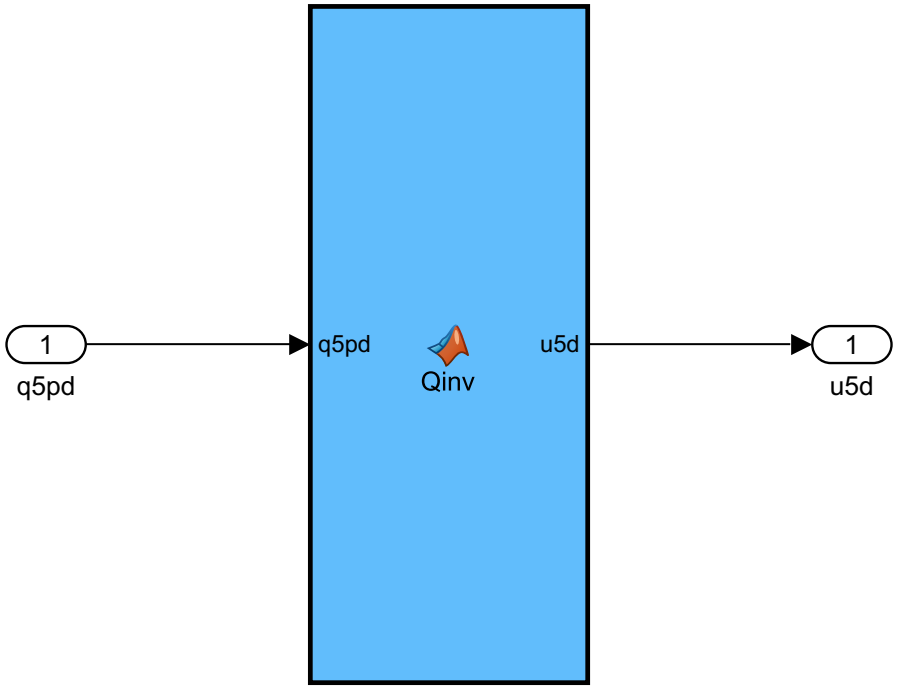


uav_con_manipulador_2d/Control traslación/D	7



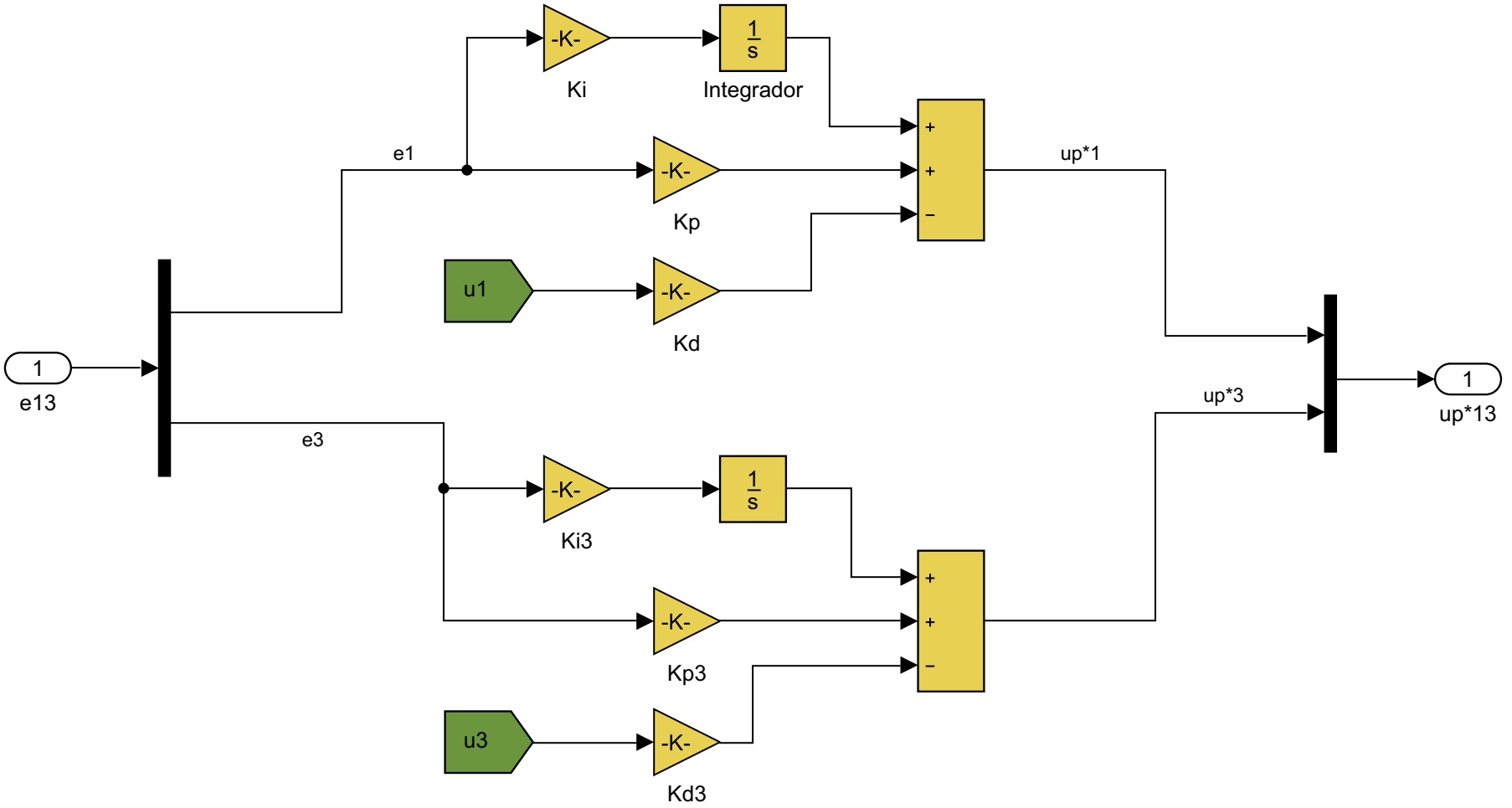
uav_con_manipulador_2d/Control traslación/F123 inv	8

```
function [F3d,q5d] = F123inv(u1pd,u3pd,parametros)
M=parametros(1);
g=parametros(2);

F3d=M*sqrt(u1pd^2+(u3pd+g)^2);
q5d=atan2(M*(u1pd/F3d),sqrt(1-M*(u1pd/F3d)));
```

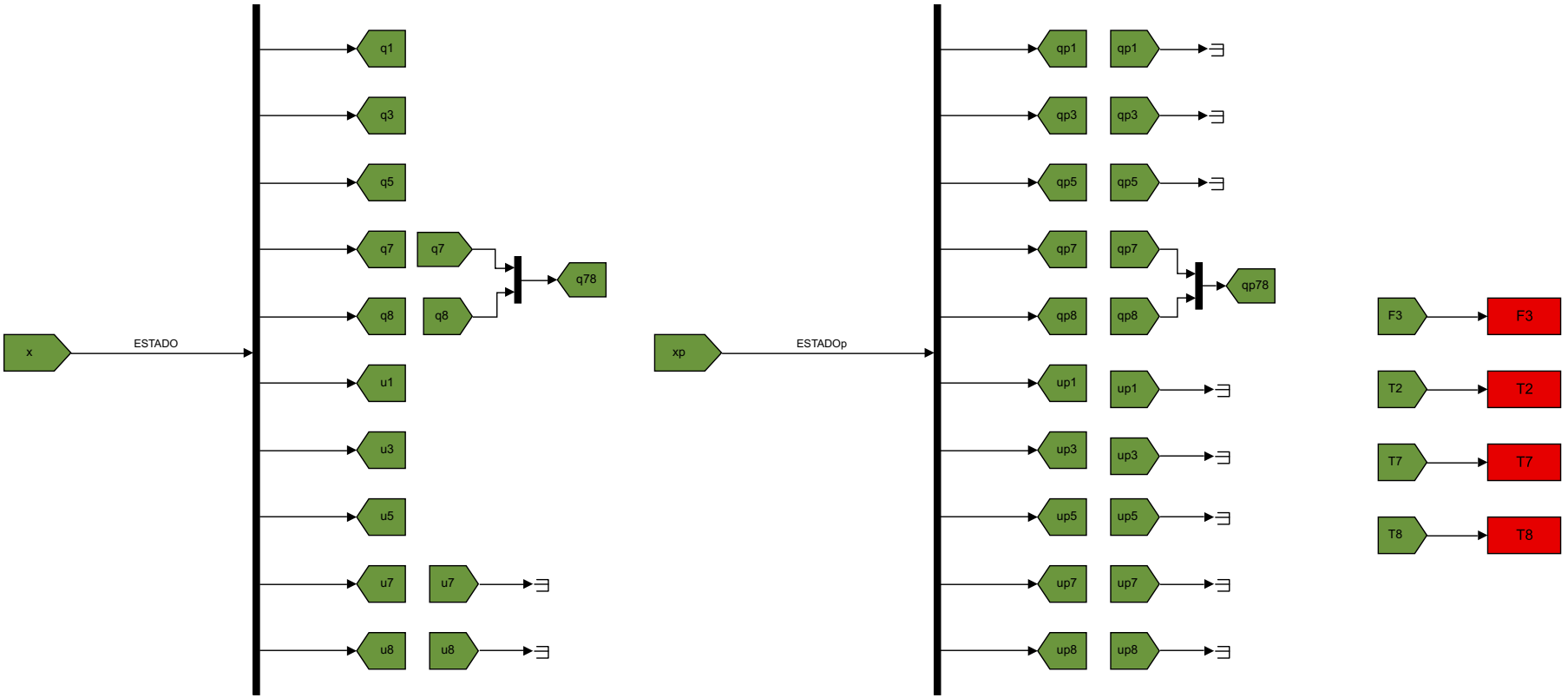


uav_con_manipulador_2d/Control traslación/Q inv	10

```
function u5d  = Qinv(q5pd)

u5d=q5pd;
```

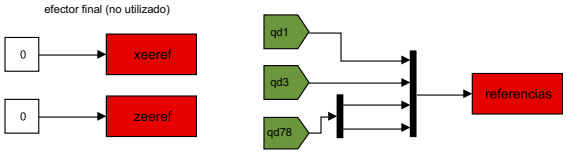


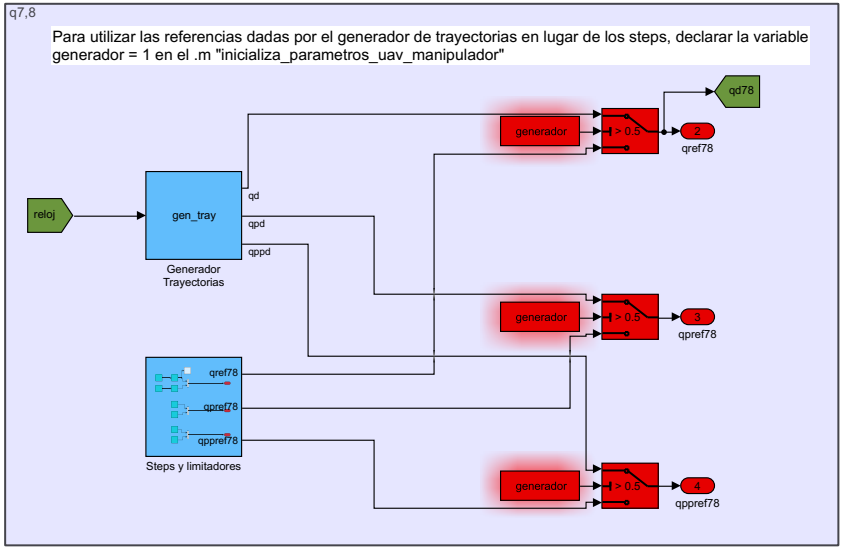
uav_con_manipulador_2d/Control traslación/Rtrans	12

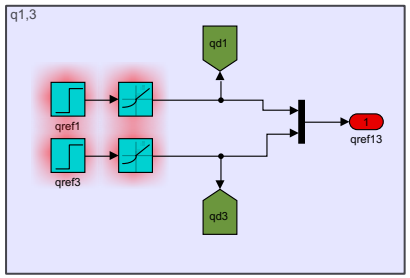

x
ESTADO
q1
q3
q5
q7
q7
q78
q8
q8
u1
u3
u5
u7
u7
u8
u8
xp
ESTADOp
qp1
qp1
qp3
qp3
qp5
qp5
qp7
qp7
qp78
qp8
qp8
up1
up1
up3
up3
up5
up5
up7
up7
up8
up8
F3
F3
T2
T2
T7
T7
T8
T8

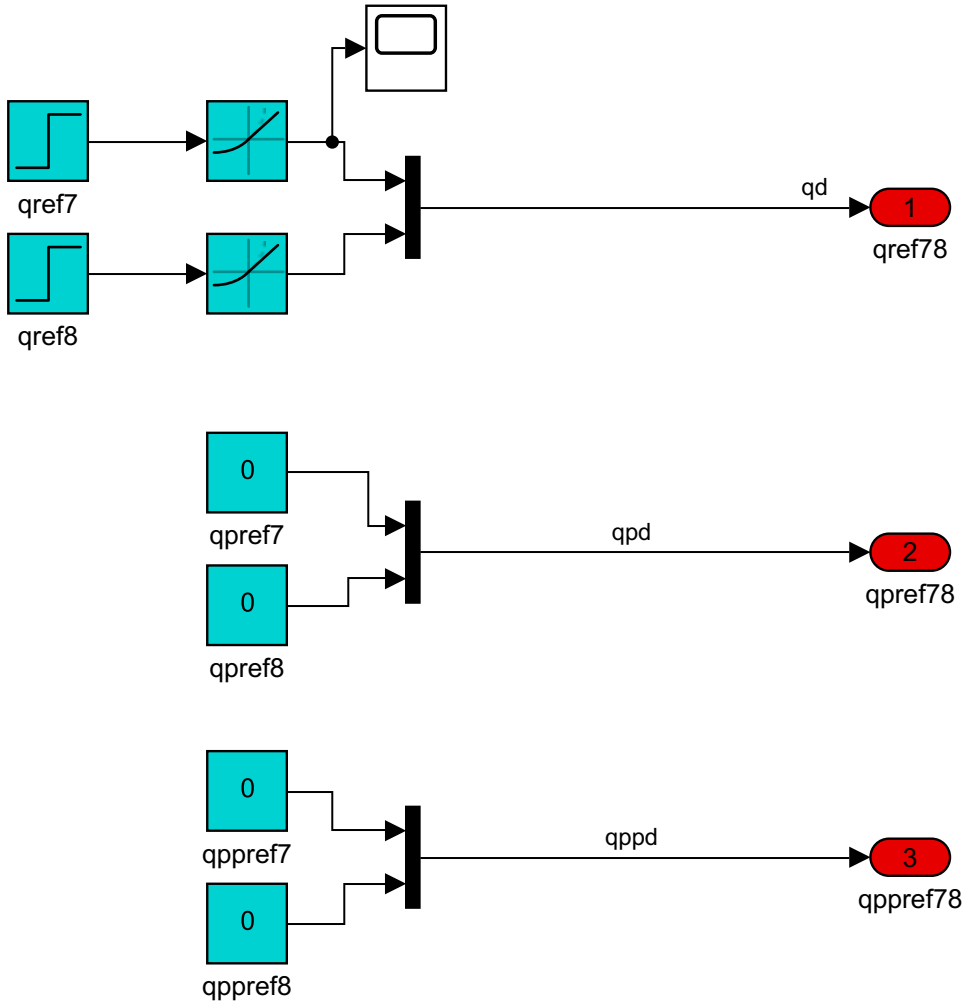
```
function xp = modelodin(FR3,T2,T7,T8,x,parametros)

xp  = zeros(10,1);
COEF = zeros(5,5);
RHS = zeros(1,5);

%ENTRADAS

q1=x(1); q3=x(2); q5=x(3); q7=x(4);q8=x(5);
u1=x(6); u3=x(7); u5=x(8); u7=x(9); u8=x(10);

M=parametros(1);
g=parametros(2);
I11=parametros(3);
I22=parametros(4);
I33=parametros(5);
l1  = parametros(6);
l2  = parametros(7);
m1  = parametros(8);
m2  = parametros(9);
d   = parametros(10);

% Ecuaciones cinemáticas:
q1p = u1;
q3p = u3;
q5p = u5;
q7p = u7;
q8p = u8;

% Ecuaciones dinámicas:
COEF(1,1) = -M - m1 - m2;
COEF(1,2) = 0;
COEF(1,3) = m1*(d*cos(q5)+l1*sin(q5+q7)) + m2*(d*cos(q5)+l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(1,4) = l1*m1*sin(q5+q7) + m2*(l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(1,5) = l2*m2*sin(q5+q7+q8);
COEF(2,1) = 0;
COEF(2,2) = -M - m1 - m2;
COEF(2,3) = -m1*(d*sin(q5)-l1*cos(q5+q7)) - m2*(d*sin(q5)-l1*cos(q5+q7)-l2*cos(q5+q7+q8));
COEF(2,4) = l1*m1*cos(q5+q7) + m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8));
COEF(2,5) = l2*m2*cos(q5+q7+q8);
COEF(3,1) = m1*(d*cos(q5)+l1*sin(q5+q7)) + m2*(d*cos(q5)+l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(3,2) = -m1*(d*sin(q5)-l1*cos(q5+q7)) - m2*(d*sin(q5)-l1*cos(q5+q7)-l2*cos(q5+q7+q8));
COEF(3,3) = -I22 - m1*(d^2+l1^2+2*d*l1*sin(q7)) - m2*(d^2+l1^2+l2^2+2*d*l1*sin(q7)+2*l1*l2*cos(q8)+2*d*l2*sin(q7+q8));
COEF(3,4) = -l1*m1*(l1+d*sin(q7)) - m2*(l1^2+l2^2+d*l1*sin(q7)+2*l1*l2*cos(q8)+d*l2*sin(q7+q8));
COEF(3,5) = -l2*m2*(l2+l1*cos(q8)+d*sin(q7+q8));
COEF(4,1) = l1*m1*sin(q5+q7) + m2*(l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(4,2) = l1*m1*cos(q5+q7) + m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8));
COEF(4,3) = -l1*m1*(l1+d*sin(q7)) - m2*(l1^2+l2^2+d*l1*sin(q7)+2*l1*l2*cos(q8)+d*l2*sin(q7+q8));
COEF(4,4) = -m1*l1^2 - m2*(l1^2+l2^2+2*l1*l2*cos(q8));
COEF(4,5) = -l2*m2*(l2+l1*cos(q8));
COEF(5,1) = l2*m2*sin(q5+q7+q8);
COEF(5,2) = l2*m2*cos(q5+q7+q8);
COEF(5,3) = -l2*m2*(l2+l1*cos(q8)+d*sin(q7+q8));
COEF(5,4) = -l2*m2*(l2+l1*cos(q8));
COEF(5,5) = -m2*l2^2;
RHS(1) = m1*(d*sin(q5)*u5^2-l1*cos(q5+q7)*(u5+u7)^2) + m2*(d*sin(q5)*u5^2-l1*cos(q5+q7)*(u5+u7)^2-l2*cos(q5+q7+q8)*(u5+u7+u8)^2) - FR3*sin(
RHS(2) = g*M + g*m1 + g*m2 + m1*(d*cos(q5)*u5^2+l1*sin(q5+q7)*(u5+u7)^2) + m2*(d*cos(q5)*u5^2+l1*sin(q5+q7)*(u5+u7)^2+l2*sin(q5+q7+q8)*(u5+
```

```
RHS(3) = g*m1*(d*sin(q5)-l1*cos(q5+q7)) + g*m2*(d*sin(q5)-l1*cos(q5+q7)-l2*cos(q5+q7+q8)) - T2 - d*l1*m1*cos(q7)*(u5^2-(u5+u7)^2) - m2*(d*l
RHS(4) = m2*(l1*l2*sin(q8)*(u5+u7)^2-d*l1*cos(q7)*u5^2-d*l2*cos(q7+q8)*u5^2-l1*l2*sin(q8)*(u5+u7+u8)^2) - T7 - g*l1*m1*cos(q5+q7) - g*m2*(l
RHS(5) = -T8 - g*l2*m2*cos(q5+q7+q8) - l2*m2*(d*cos(q7+q8)*u5^2-l1*sin(q8)*(u5+u7)^2);
SolutionToAxEqualsB = COEF\RHS';

% Update variables after uncoupling equations
u1p = SolutionToAxEqualsB(1);
u3p = SolutionToAxEqualsB(2);
u5p = SolutionToAxEqualsB(3);
u7p = SolutionToAxEqualsB(4);
u8p = SolutionToAxEqualsB(5);

% Output:
xp(1)  = q1p;
xp(2)  = q3p;
xp(3)  = q5p;
xp(4)  = q7p;
xp(5)  = q8p;
xp(6)  = u1p;
xp(7)  = u3p;
xp(8)  = u5p;
xp(9)  = u7p;
xp(10) = u8p;
```


efector final (no utilizado)
0
xeeref
0
zeeref
qd1
qd3
qd78
referencias


q7,8
Para utilizar las referencias dadas por el generador de trayectorias en lugar de los steps, declarar la variable generador = 1 en el .m "inicializa_parametros_uav_manipulador"
reloj
gen_tray
qd
qpd
qppd
Generador Trayectorias
generador
> 0.5
2
qref78
qd78
generador
> 0.5
3
qpref78
qref78
qpref78
qppref78
Steps y limitadores
generador
> 0.5
4
qppref78


q1,3
qd1
qref1
qref3
1
qref13
qd3



uav_con_manipulador_2d/Referencias/Steps y limitadores	16

ANEXO G

- **Programa que contiene la declaración de las variables generales de la simulación.**

```
%PARÁMETROS DEL MODELO DE SIMULINK
clear all
%Nota: Todo en unidades del sistema internacional.

%% SIMULACIÓN_____________________________________
t_sim=60;
dt=0.001; %Paso de simulación

%Condiciones iniciales
q10=0;
q30=0;
q50=0;
q70=60*pi/180;       %pi/2 es el brazo extendido hacia abajo
q80=1.25;       %
u10=0;
u30=0;
u50=0;
u70=0;
u80=0;
x0=[q10,q30,q50,q70,q80,u10,u30,u50,u70,u80];

%Las referencias están puestas como escalones con t_qrefi,
%qi0 y qrefi.
t_qref1=10;
t_qref3=20;
t_qref7=30;
t_qref8=40;

qref1=2;
qref3=3;
qref7=q70+0.8;%20*pi/180;%pi/2;% +0.5236; %90+30 °
qref8=q80+0.3;%10*pi/180;%q80;%pi/3;

%posiciones del efector final deseadas
% xrefee=0;
% zrefee=0;
% t_xrefee=10;
% t_zrefee=30;
% xrefee=0;
% zrefee=0;
% z_offset=0.7;%la distancia a la que se sitúa la plataforma en vertical

%% CONTROLADORES_____________________________________
p=-3; %-4 esta ok, -1 es el orginal
p1 = p;   p2 = p;   p3 = p;   p4 = p;   p5 = p;

%Rori
Kp_q5= p4*p5 + (p4+p5)*p3 + (p3+p4+p5)*p2 + (p2+p3+p4+p5)*p1;

Kp_u5= -(p1+p2+p3+p4+p5);

%Rtrans
Kp_q1=(p2*p3*p4*p5 + p1*(p3*p4*p5 + p2*(p3*(p4+p5) + p4*p5)))/Kp_q5;
Ki_q1= -p1*p2*p3*p4*p5/Kp_q5;
Kd_q1=-(p3*p4*p5 + p2*(p3*(p4+p5) + p4*p5) + p1*(p4*p5 + p3*(p4+p5) +...
          (p3+p4+p5)*p2))/Kp_q5;
```

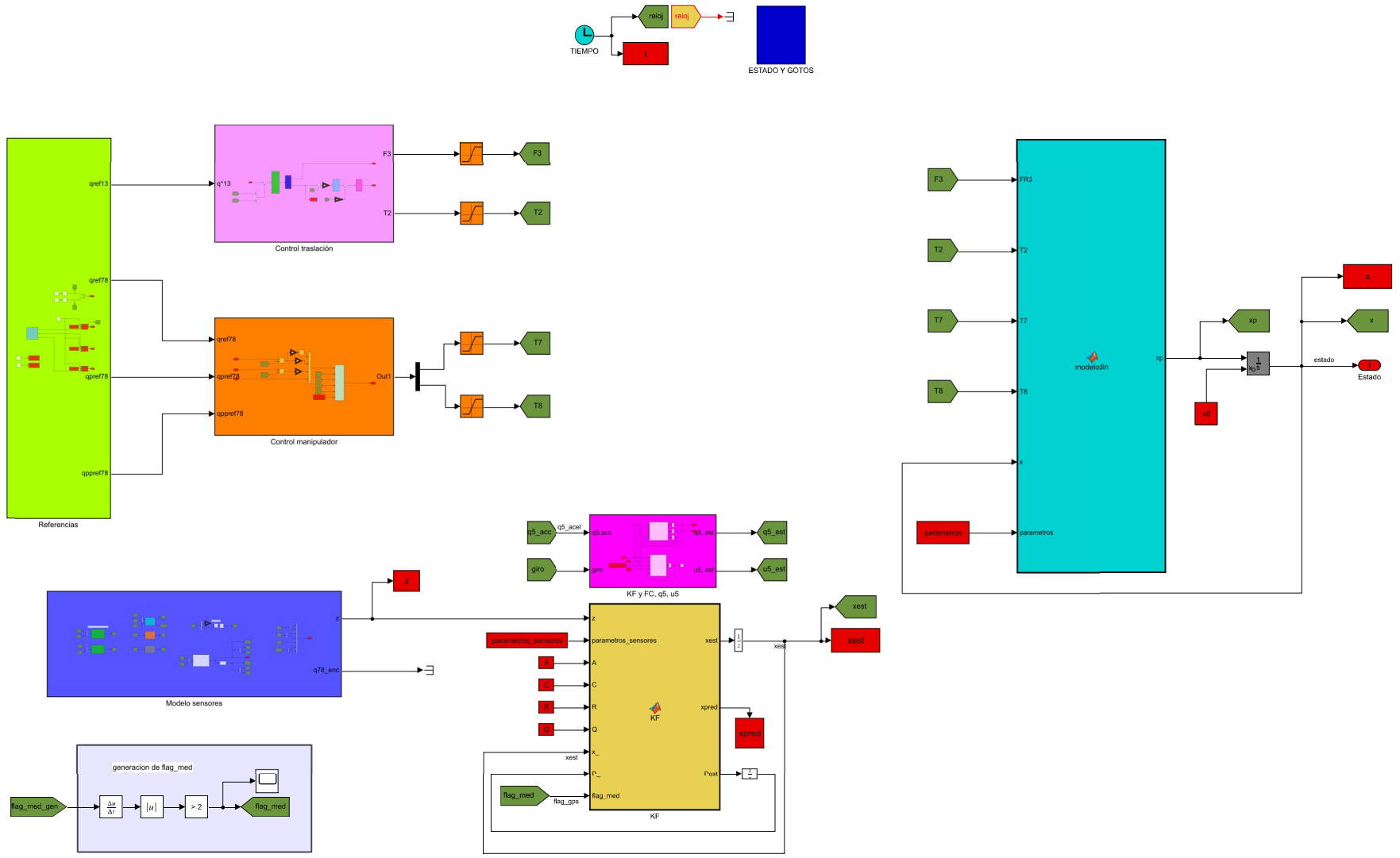
```
Kp_q3 =   p1*(p2+p3)+p2*p3;
Ki_q3 = - p1*p2*p3;
Kd_q3 = -(p1+p2+p3);

%% DINÁMICA_____________________________________________
I11 = 0.082;
I22 = 0.082;
I33 = 0.149;
M=4;
g=9.81;

%% SATURACIÓN DE LA ACTUACIÓN_____________________________
tmax=20;
F3_max=200;
F3_min=0;
T2_max=tmax;
T2_min=-tmax;
T7_max=tmax;
T7_min=-tmax;
T8_max=tmax;
T8_min=-tmax;

%% PARAMETROS DEL BRAZO MANIPULADOR
  L1=0.2;%era 0.3
  L2=0.15;%era 0.3

  m1=0.1;%era 0.15
  m2=0.1;%era 0.15

  d= 0;

  i11=m1*L1^2;
  i22=m2*L2^2;

%Parametros de los motores
  R1=1;
  R2=1;

  Jm1=0;%0.1*i11;
  Jm2=0;%0.1*i22;

  Bm1=0;%1e-7;
  Bm2=0;%1e-7;

%% GENERADOR DE TRAYECTORIAS
generador=0;%1 -> trayectorias por generador alternativo
            %0 -> trayectorias con steps y limitadores


%% BRAZO MANIPULADOR
  Kp_q7=8;
  Kd_q7=4;%15
  Ki_q7=0.5;

  Kp_q8=15;
  Kd_q8=4;
  Ki_q8=0.5;
```

```
%% FIN_____________________________________________________
parametros=[M,g,I11,I22,I33,L1,L2,m1,m2,d,i11,i22,R1,R2,Jm1,Jm2,Bm1,Bm2];
```

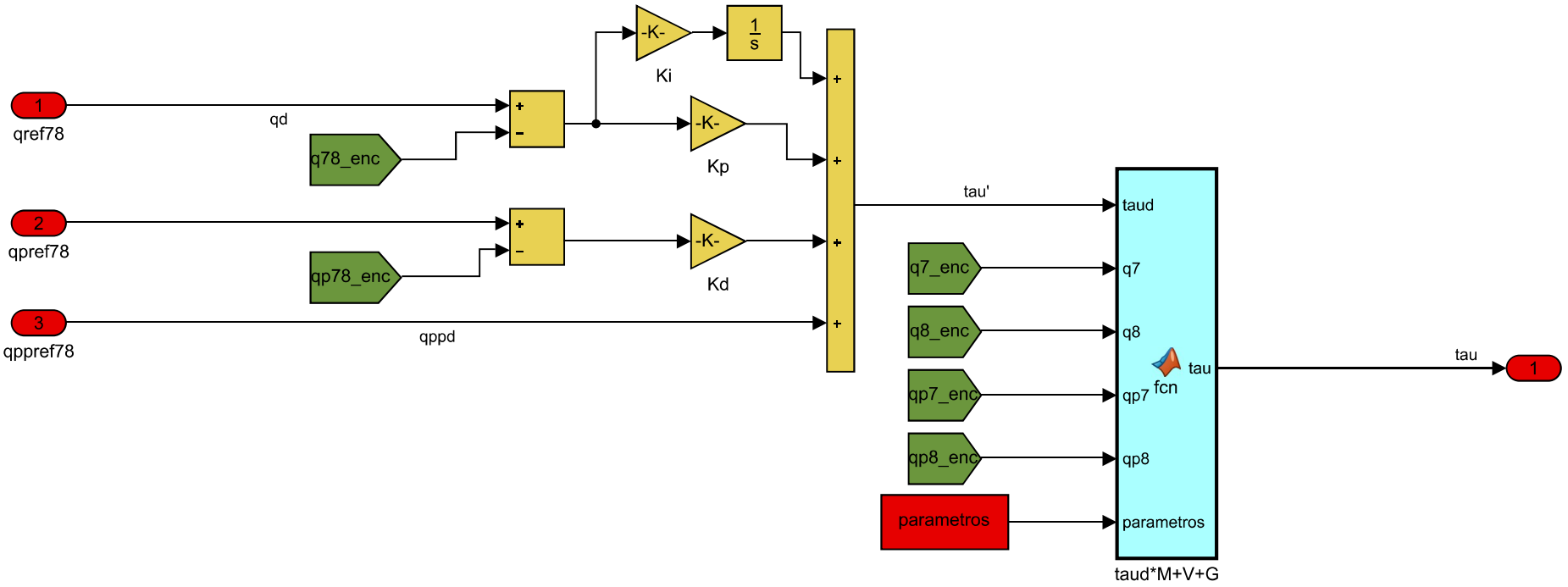
- **Programa que contiene la declaración de las variables generales de los sensores.**

```
%_________________________________________


%% Notas_____________________________________
% select = 0 -> modelo de HF + LF
% select = 1 -> modelo de espacio de estados
%_________________________________________
sel=1;

%% GPS
select_gps=sel;
f_gps=10;       %Hz
deltat_gps=1/f_gps;

sigma_gpsp=2.5;   %m
sigma_gpsv=0.05;  %m/s

% N_gpsp=[sigma_px*randn; sigma_py*randn; sigma_pz*randn];
%
% N_gpsv=[sigma_vx*randn; sigma_vy*randn; sigma_vz*randn];

%% Acelerómetro
select_acc=sel;
f_acc=100;      %Hz
deltat_acc=1/f_acc;

sigma_acc=9.8*1e-3; %m/s^2 -rms-


b1_acc=0;
b2_acc=0;
b3_acc=0;

% N_acc=sigma_acc.*[randn;randn;randn];
%
% b_acc=[b1_acc; b2_acc; b3_acc];

%% Giróscopo
select_giro=sel;
f_giro=100;     %Hz
deltat_giro=1/f_giro;

sigma_giro=0.04*pi/180; %0.04 dps -rms-

b1_giro=0;
b2_giro=0;
b3_giro=0;

% N_giro=sigma_giro.*[randn;randn;randn];
%
% b_giro=[b1_giro; b2_giro; b3_giro];


%% Filtro complementario
```

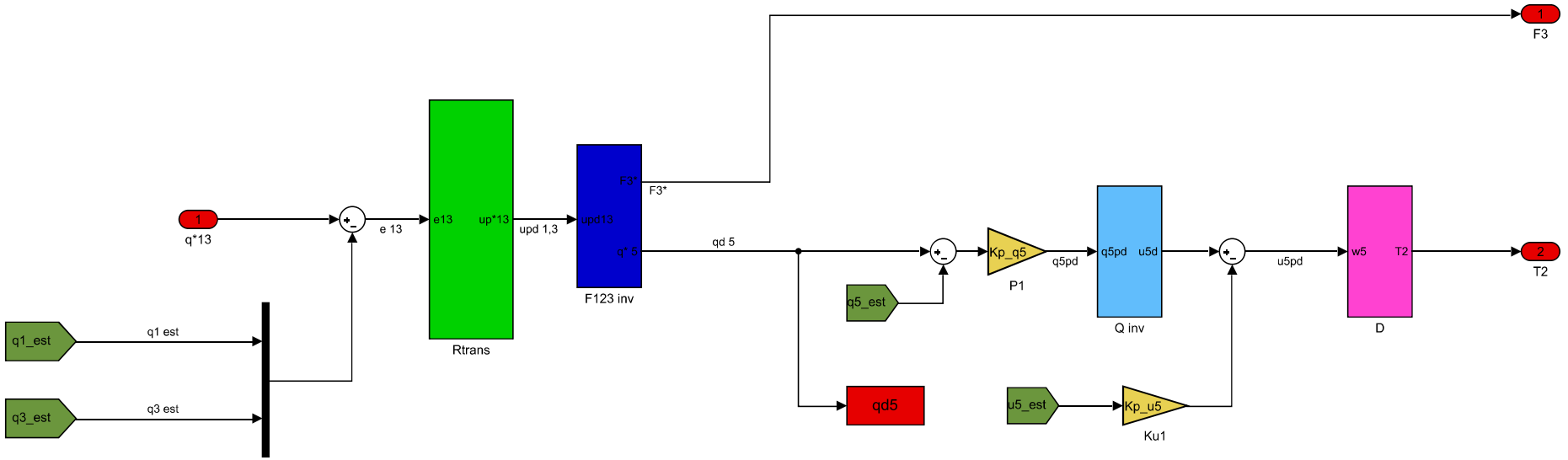
```
alfa=0.2;
sigma_fc=0;

%% Altímetro barométrico
select_bar=sel;
f_bar=10;       %Hz
deltat_bar=1/f_bar;

p0=1013.25; %hPa
T0=288.15;  %K

sigma_bar=0.4;   %m


%% EKF

%%Inicializaciones
A=[1,   0,  dt, 0,  0.5*dt^2,   0   ;...
   0,   1,  0,  dt,    0,   0.5*dt^2;...
   0,   0,  1,  0,    dt,   0   ;...
   0,   0,  0,  1,     0,   dt  ;...
   0,   0,  0,  0,     1,   0   ;...
   0,   0,  0,  0,     0,   1   ];     %6x6

B=0;

C=[1,   0,  0,  0,  0,  0;...
   0,   1,  0,  0,  0,  0;...
   0,   0,  1,  0,  0,  0;...
   0,   0,  0,  1,  0,  0;...
   0,   1,  0,  0,  0,  0;...
   0,   0,  0,  0,  1,  0;...
   0,   0,  0,  0,  0,  1];

trust=0.001;

R=(trust^2).*eye(6);%confianza en la prediccion  %6x6

Q=[sigma_gpsp^2,    0,  0,  0,  0,  0,  0;...    %7x7
   0,   sigma_gpsp^2,   0,  0,  0,  0,  0;...
   0,   0,  sigma_gpsv^2,   0,  0,  0,  0;...
   0,   0,  0,  sigma_gpsv^2,   0,  0,  0;...
   0,   0,  0,  0,  sigma_bar^2,    0,  0;...
   0,   0,  0,  0,  0,  sigma_acc^2,    0;...
   0,   0,  0,  0,  0,  0,  sigma_acc^2 ];

%% Encoders
select_enc=sel;
f_enc=1/dt; %Frec. max. ~250 KHz
deltat_enc=dt;

sigma_enc= 0.3*pi/180; %0.3°


%% Salida
%Componente-------------1----------2-----------3---------4----------5--------
--6--------7-----------8----------9--------10-----11---121314,1516
parametros_sensores=[sigma_gpsp,sigma_gpsv,sigma_acc,sigma_giro,sigma_bar,del
tat_gps,deltat_acc,deltat_giro,deltat_bar,alfa,sigma_fc];%,A,B,C,R,Q];
```

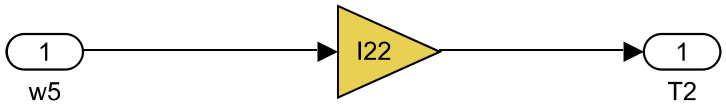


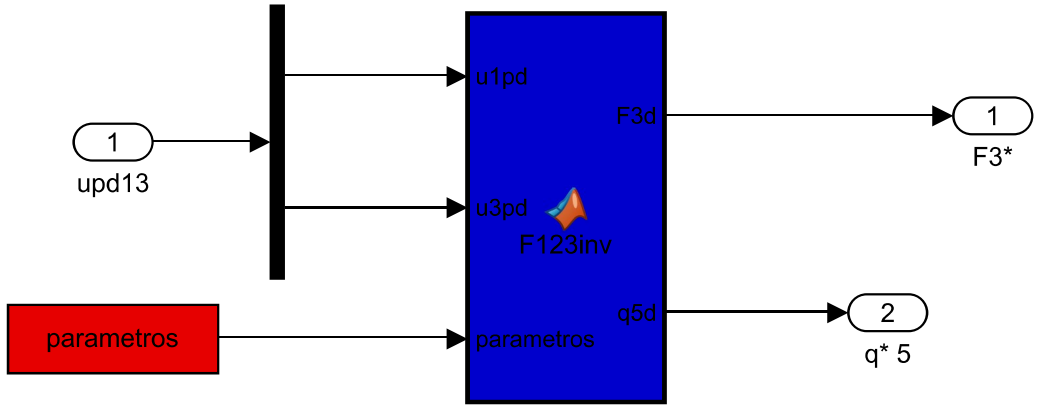
uav_con_manipulador_con_sensores_2d_Q5KF	1


1
qref78
qd
q78_enc
-K-
Ki
1
s
-K-
Kp
2
qpref78
qp78_enc
-K-
Kd
3
qppd
qppref78
tau'
taud
q7_enc
q7
q8_enc
q8
qp7_enc
qp7
qp8_enc
qp8
parametros
parametros
fcn
tau
taud*M+V+G
tau
1

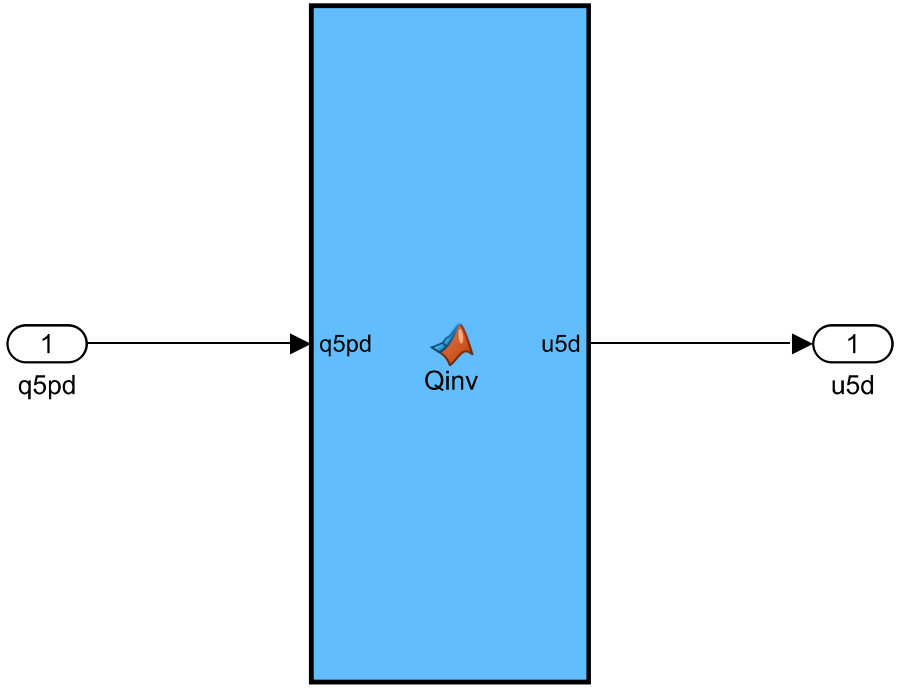
```
function tau = fcn(taud,q7,q8,qp7,qp8,parametros)

%% Inicializacion
tau=[0;0];

%% Parámetros
%             1,2,3,,,,4,,5,,,6,,7,,8,,9,,10,11,12,,13,14,15,,16,,17,,18
% parametros=[M,g,I11,I22,I33,L1,L2,m1,m2,d,i11,i22,R1,R2,Jm1,Jm2,Bm1,Bm2];
g=parametros(2);
l1=parametros(6);
l2=parametros(7);
m1=parametros(8);
m2=parametros(9);
% i11=parametros(11);
% i22=parametros(12);
% R1=parametros(13);
% R2=parametros(14);
% Jm1=parametros(15);
% Jm2=parametros(16);
% Bm1=parametros(17);
% Bm2=parametros(18);

%% Cálculo de matrices y vectores
M=[l2^2*m2+2*l1*l2*m2*cos(q8)+l1^2*(m1+m2) , l2^2*m2+l1*l2*m2*cos(q8);...
   l2^2*m2+l1*l2*m2*cos(q8) , l2^2*m2];

V=[-m2*l1*l2*sin(q8)*qp8^2-2*m2*l1*l2*sin(q8)*qp7*qp8;...
   m2*l1*l2*sin(q8)*qp7^2];

G=(-1).*[m2*l2*g*cos(q7+q8)+(m1+m2)*l1*g*cos(q7);...
         m2*l2*g*cos(q7+q8)];


%% Salida
tau=M*taud + V + G;
% tau=taud;   %Usado en muchos controles industrialessegun J.Craig
```


1
q*13
e 13
e13
up*13
Rtrans
upd 1,3
F3*
upd13
q* 5
F123 inv
F3*
1
F3
qd 5
q1_est
q1 est
q3_est
q3 est
q5_est
Kp_q5
P1
q5pd
q5pd
u5d
Q inv
u5pd
w5
T2
D
2
T2
qd5
u5_est
Kp_u5
Ku1


1
w5
I22
1
T2

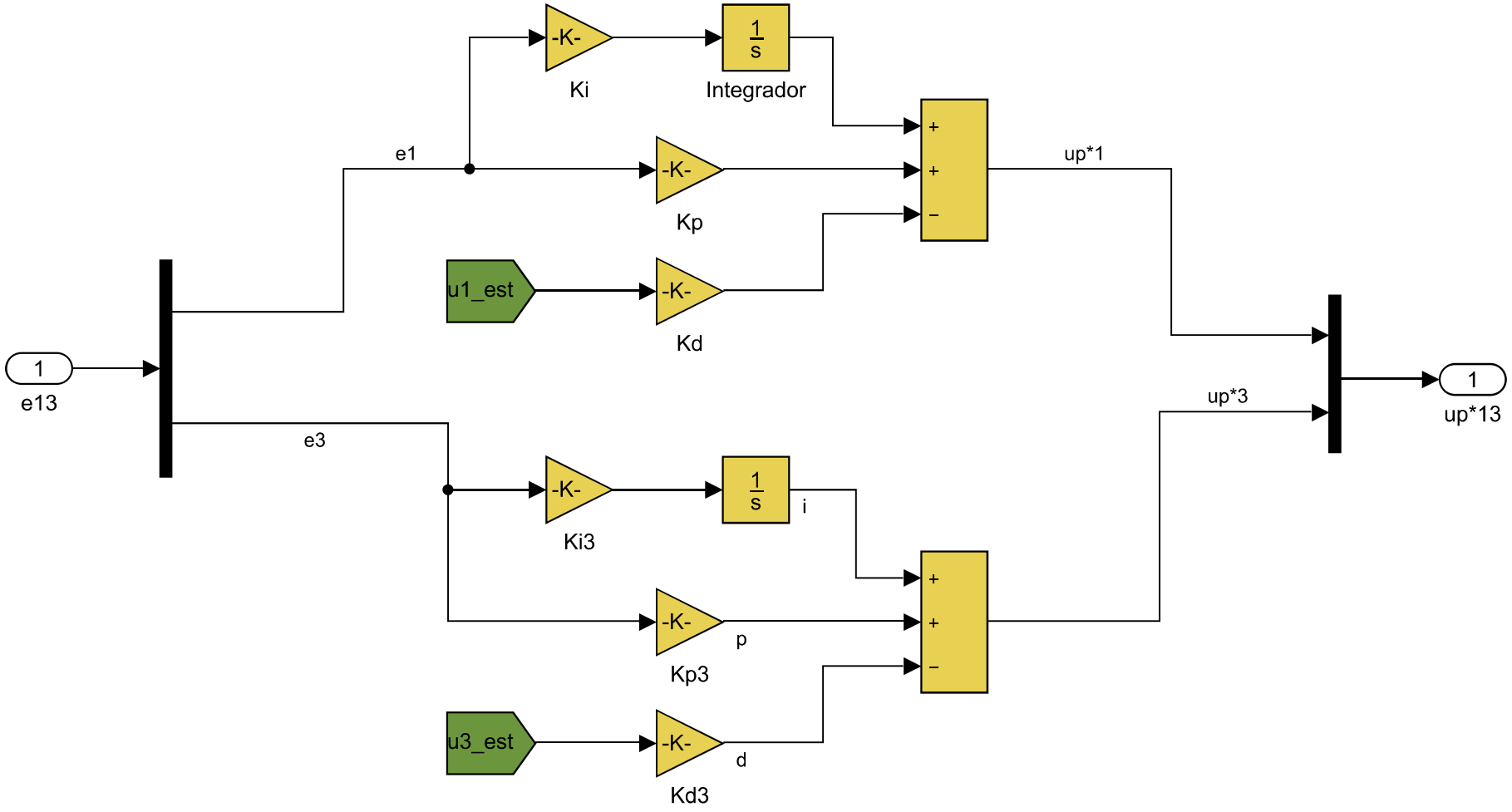

1
upd13
parametros
u1pd
u3pd
parametros
F3d
q5d
F123inv
1
F3*
2
q* 5

```
function [F3d,q5d] = F123inv(u1pd,u3pd,parametros)
M=parametros(1);
g=parametros(2);

F3d=M*sqrt(u1pd^2+(u3pd+g)^2);
q5d=atan2(M*(u1pd/F3d),sqrt(1-M*(u1pd/F3d)));
```

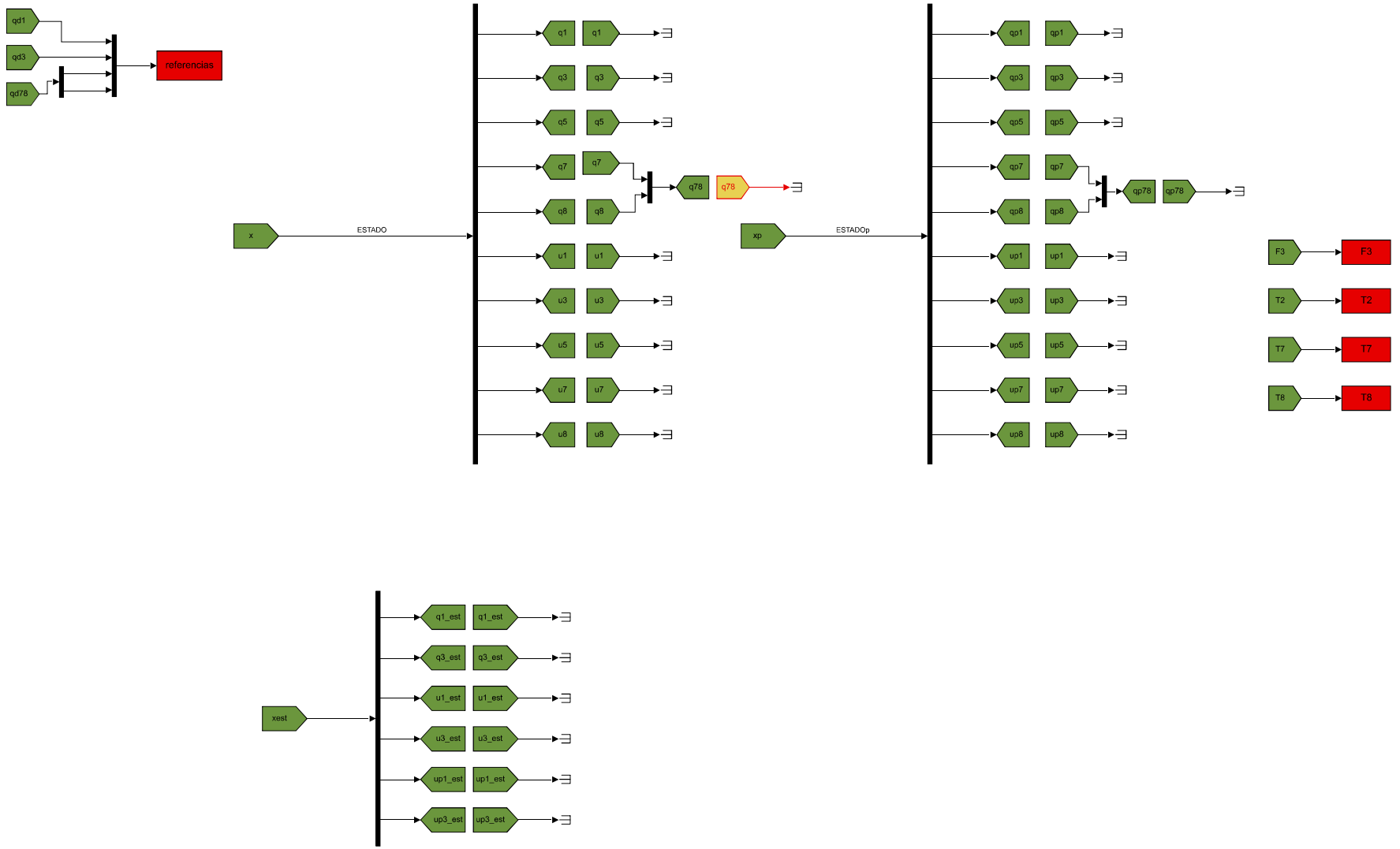


uav_con_manipulador_con_sensores_2d_Q5KF/Control traslación/Q inv	8

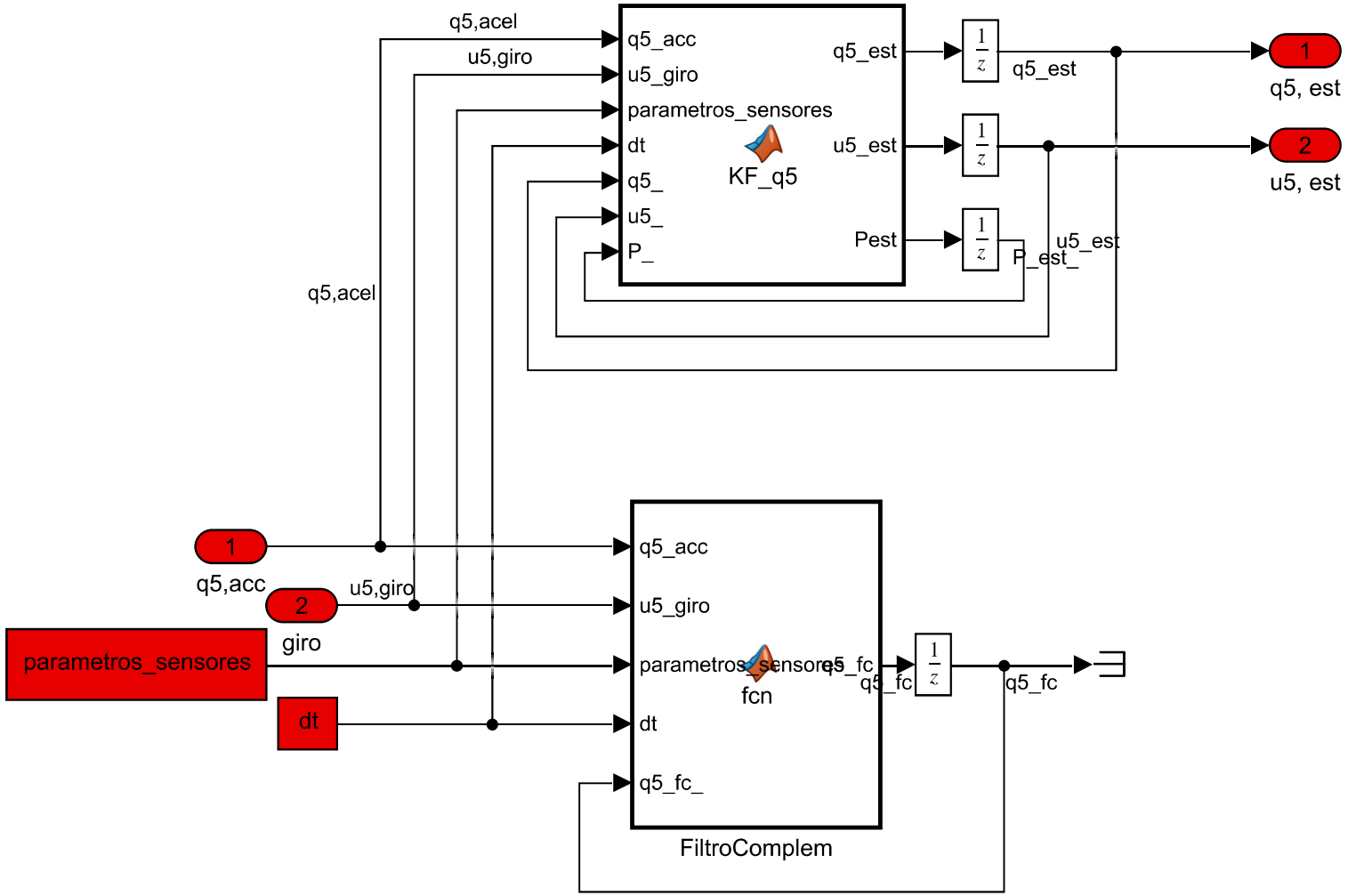
```
function u5d  = Qinv(q5pd)

u5d=q5pd;
```



uav_con_manipulador_con_sensores_2d_Q5KF/Control traslación/Rtrans	10


qd1
qd3
qd78
referencias
x
ESTADO
q1
q3
q5
q7
q8
u1
u3
u5
u7
u8
q78
xp
ESTADOp
qp1
qp3
qp5
qp7
qp8
up1
up3
up5
up7
up8
qp78
F3
T2
T7
T8
xest
q1_est
q3_est
u1_est
u3_est
up1_est
up3_est

```
function [xest,xpred,Pest]= KF(z,parametros_sensores,A,C,R,Q,x_,P_,flag_med)
%Implementación de un EKF para estimación de estado

%% Inicialización

xpred=zeros(6,1);
xest=zeros(6,1);
Ppred=zeros(6,6);
Pest=zeros(6,6);

%% FASE DE PREDICCIÓN (a la frecuencia del paso de simulación)

xpred=A*x_;

Ppred=A*P_*A' + R;

%% Fase de corrección (a la freciencia de medidas de GPS y BAR)

if flag_med==1

    K=Ppred*C' / (C*Ppred*C' + Q);

    xest=xpred+K*(z-C*xpred);

    Pest=(eye(6)-K*C)*Ppred;

else

    xest=xpred;

    Pest=Ppred;

end
```

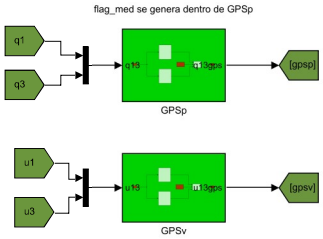

q5,acel
u5,giro
q5_acc
u5_giro
parametros_sensores
dt
q5_
u5_
P_
q5_est
u5_est
Pest
KF_q5
1/z
q5_est
u5_est
P_est_
1
q5, est
2
u5, est
q5,acel
1
q5,acc
2
giro
u5,giro
parametros_sensores
dt
q5_acc
u5_giro
parametros_sensores
dt
q5_fc_
fcn
q5_fc
q5_fc
FiltroComplem

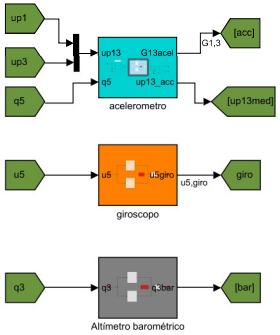
```
function q5_fc = fcn(q5_acc,u5_giro,parametros_sensores,dt,q5_fc_)
q5_fc=0;
% deltat_giro=parametros_sensores(8);
alfa=parametros_sensores(10);

q5_fc=(1-alfa)*(q5_fc_ + u5_giro*dt) + alfa*q5_acc;
```

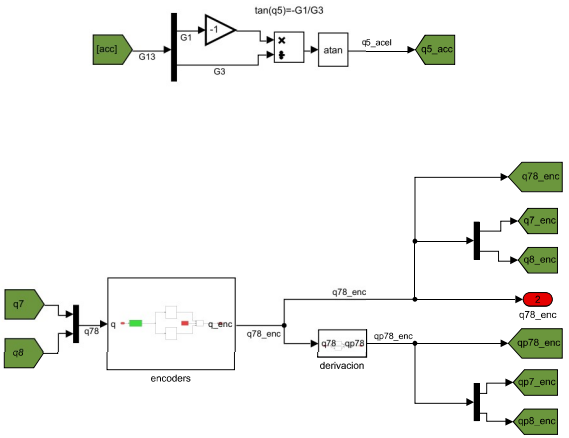
```
function [q5_est,u5_est,Pest] = KF_q5(q5_acc,u5_giro,parametros_sensores,dt,q5_,u5_,P_)

%% Inicialización

xpred=zeros(2,1);
xest=zeros(2,1);
Ppred=zeros(2,2);
Pest=zeros(2,2);
q5_est=0;
u5_est=0;

%Ins
x_=[q5_,u5_]';
z=[q5_acc,u5_giro]';

% sigma_acc=parametros_sensores(3);
sigma_q5acc=0.313;
sigma_giro=parametros_sensores(4);


A=[1,dt ; 0,1];
C=eye(2);

trust_q5=0.0001;
R=trust_q5^2.*eye(2);
Q=[sigma_q5acc^2,0;0,sigma_giro^2]; %OJO A LA SIGMA DEL ACC


%% FASE DE PREDICCIÓN (a la frecuencia del paso de simulación)

xpred=A*x_;

Ppred=A*P_*A' + R;

%% Fase de corrección

K=Ppred*C' / (C*Ppred*C' + Q);

xest=xpred+K*(z-C*xpred);

Pest=(eye(2)-K*C)*Ppred;


q5_est=xest(1);
u5_est=xest(2);

end
```

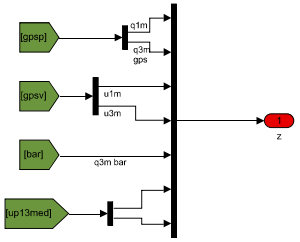
```
function xp = modelodin(FR3,T2,T7,T8,x,parametros)
%cuidado con el nombre de F3
xp  = zeros(10,1);
COEF = zeros(5,5);
RHS = zeros(1,5);

%ENTRADAS

q1=x(1); q3=x(2); q5=x(3); q7=x(4);q8=x(5);
u1=x(6); u3=x(7); u5=x(8); u7=x(9); u8=x(10);

M=parametros(1);
g=parametros(2);
I11=parametros(3);
I22=parametros(4);
I33=parametros(5);
l1  = parametros(6);
l2  = parametros(7);
m1  = parametros(8);
m2  = parametros(9);
d   = parametros(10);

% Kinematic equations:
q1p = u1;
q3p = u3;
q5p = u5;
q7p = u7;
q8p = u8;

% Dynamics equations:
COEF(1,1) = -M - m1 - m2;
COEF(1,2) = 0;
COEF(1,3) = m1*(d*cos(q5)+l1*sin(q5+q7)) + m2*(d*cos(q5)+l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(1,4) = l1*m1*sin(q5+q7) + m2*(l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(1,5) = l2*m2*sin(q5+q7+q8);
COEF(2,1) = 0;
COEF(2,2) = -M - m1 - m2;
COEF(2,3) = -m1*(d*sin(q5)-l1*cos(q5+q7)) - m2*(d*sin(q5)-l1*cos(q5+q7)-l2*cos(q5+q7+q8));
COEF(2,4) = l1*m1*cos(q5+q7) + m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8));
COEF(2,5) = l2*m2*cos(q5+q7+q8);
COEF(3,1) = m1*(d*cos(q5)+l1*sin(q5+q7)) + m2*(d*cos(q5)+l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(3,2) = -m1*(d*sin(q5)-l1*cos(q5+q7)) - m2*(d*sin(q5)-l1*cos(q5+q7)-l2*cos(q5+q7+q8));
COEF(3,3) = -I22 - m1*(d^2+l1^2+2*d*l1*sin(q7)) - m2*(d^2+l1^2+l2^2+2*d*l1*sin(q7)+2*l1*l2*cos(q8)+2*d*l2*sin(q7+q8));
COEF(3,4) = -l1*m1*(l1+d*sin(q7)) - m2*(l1^2+l2^2+d*l1*sin(q7)+2*l1*l2*cos(q8)+d*l2*sin(q7+q8));
COEF(3,5) = -l2*m2*(l2+l1*cos(q8)+d*sin(q7+q8));
COEF(4,1) = l1*m1*sin(q5+q7) + m2*(l1*sin(q5+q7)+l2*sin(q5+q7+q8));
COEF(4,2) = l1*m1*cos(q5+q7) + m2*(l1*cos(q5+q7)+l2*cos(q5+q7+q8));
COEF(4,3) = -l1*m1*(l1+d*sin(q7)) - m2*(l1^2+l2^2+d*l1*sin(q7)+2*l1*l2*cos(q8)+d*l2*sin(q7+q8));
COEF(4,4) = -m1*l1^2 - m2*(l1^2+l2^2+2*l1*l2*cos(q8));
COEF(4,5) = -l2*m2*(l2+l1*cos(q8));
COEF(5,1) = l2*m2*sin(q5+q7+q8);
COEF(5,2) = l2*m2*cos(q5+q7+q8);
COEF(5,3) = -l2*m2*(l2+l1*cos(q8)+d*sin(q7+q8));
COEF(5,4) = -l2*m2*(l2+l1*cos(q8));
COEF(5,5) = -m2*l2^2;
RHS(1) = m1*(d*sin(q5)*u5^2-l1*cos(q5+q7)*(u5+u7)^2) + m2*(d*sin(q5)*u5^2-l1*cos(q5+q7)*(u5+u7)^2-l2*cos(q5+q7+q8)*(u5+u7+u8)^2) - FR3*sin(q
RHS(2) = g*M + g*m1 + g*m2 + m1*(d*cos(q5)*u5^2+l1*sin(q5+q7)*(u5+u7)^2) + m2*(d*cos(q5)*u5^2+l1*sin(q5+q7)*(u5+u7)^2+l2*sin(q5+q7+q8)*(u5+u
RHS(3) = g*m1*(d*sin(q5)-l1*cos(q5+q7)) + g*m2*(d*sin(q5)-l1*cos(q5+q7)-l2*cos(q5+q7+q8)) - T2 - d*l1*m1*cos(q7)*(u5^2-(u5+u7)^2) - m2*(d*l1
RHS(4) = m2*(l1*l2*sin(q8)*(u5+u7)^2-d*l1*cos(q7)*u5^2-d*l2*cos(q7+q8)*u5^2-l1*l2*sin(q8)*(u5+u7+u8)^2) - T7 - g*l1*m1*cos(q5+q7) - g*m2*(l1
```

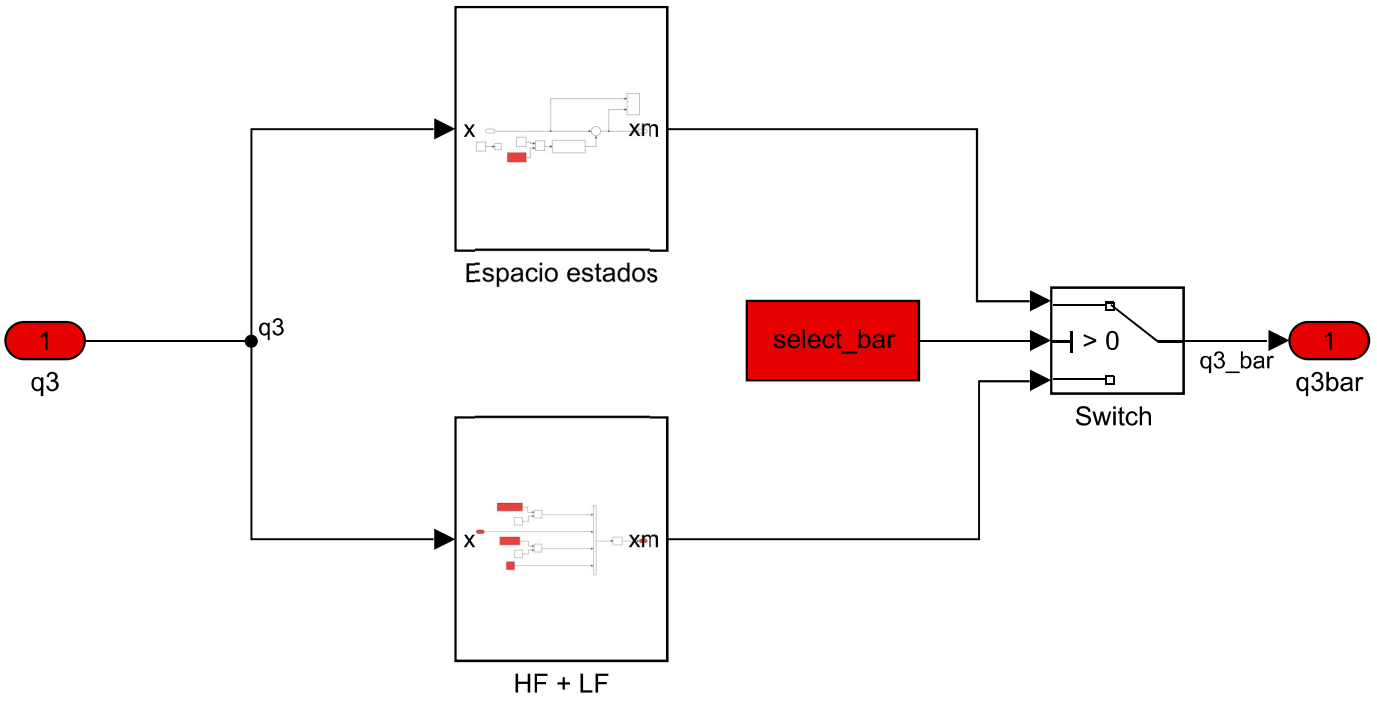
```
RHS(5) = -T8 - g*l2*m2*cos(q5+q7+q8) - l2*m2*(d*cos(q7+q8)*u5^2-l1*sin(q8)*(u5+u7)^2);
SolutionToAxEqualsB = COEF\RHS';

% Update variables after uncoupling equations
u1p = SolutionToAxEqualsB(1);
u3p = SolutionToAxEqualsB(2);
u5p = SolutionToAxEqualsB(3);
u7p = SolutionToAxEqualsB(4);
u8p = SolutionToAxEqualsB(5);

% Output:
xp(1)  = q1p;
xp(2)  = q3p;
xp(3)  = q5p;
xp(4)  = q7p;
xp(5)  = q8p;
xp(6)  = u1p;
xp(7)  = u3p;
xp(8)  = u5p;
xp(9)  = u7p;
xp(10) = u8p;
```

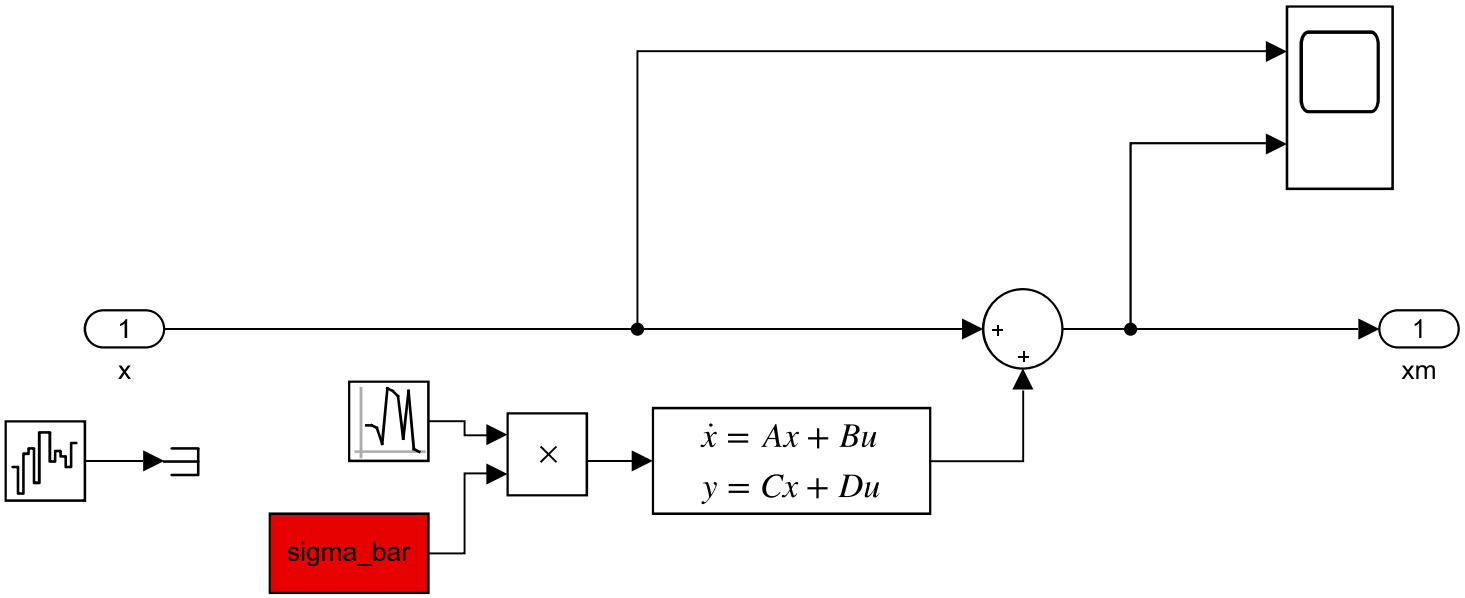

flag_med se genera dentro de GPSp
q1
q3
q13
q13gps
[gpsp]
GPSp
u1
u3
u13
u13gps
[gpsv]
GPSv

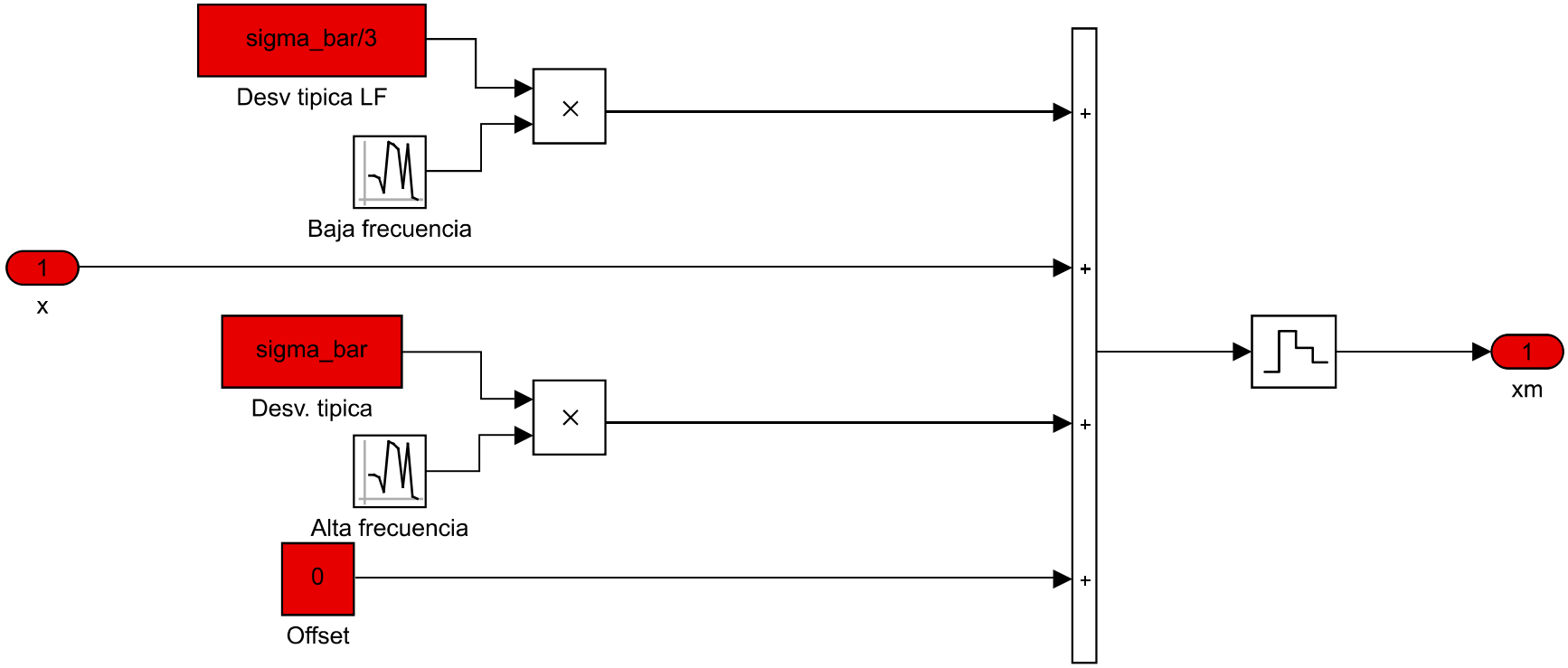

up1
up3
q5
up13
G13acel
G1,3
[acc]
up13_acc
[up13med]
acelerometro
u5
u5
u5giro
u5.giro
giro
giroscopo
q3
q3
q3bar
[bar]
Altimetro barométrico

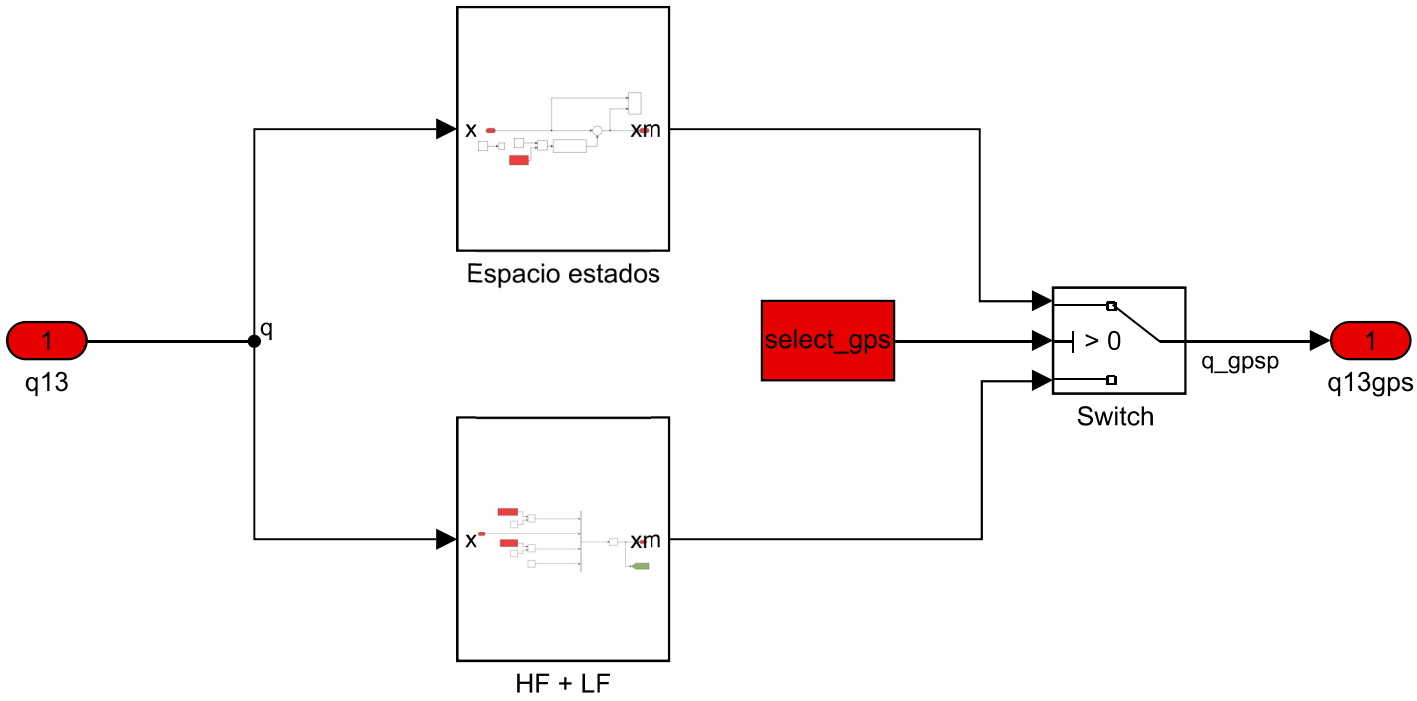

tan(q5)=-G1/G3
[acc]
G13
G1
-1
G3
atan
q5_acel
q5_acc
q7
q8
q78
q
q_enc
encoders
q78_enc
q78_enc
q78
qp78
derivacion
qp78_enc
q78_enc
q7_enc
q8_enc
2
q78_enc
qp78_enc
qp7_enc
qp8_enc

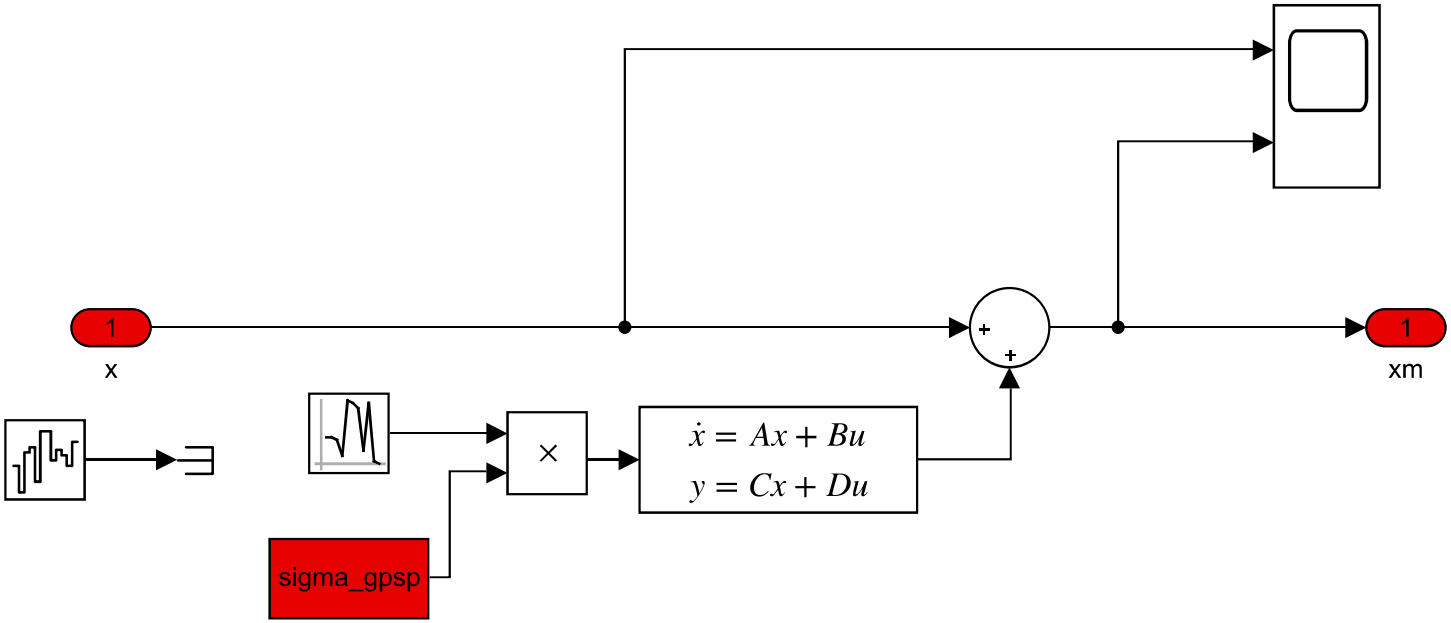

[gpsp]
q1m
q3m
gps
[gpsv]
u1m
u3m
[bar]
q3m bar
[up13med]
1
z

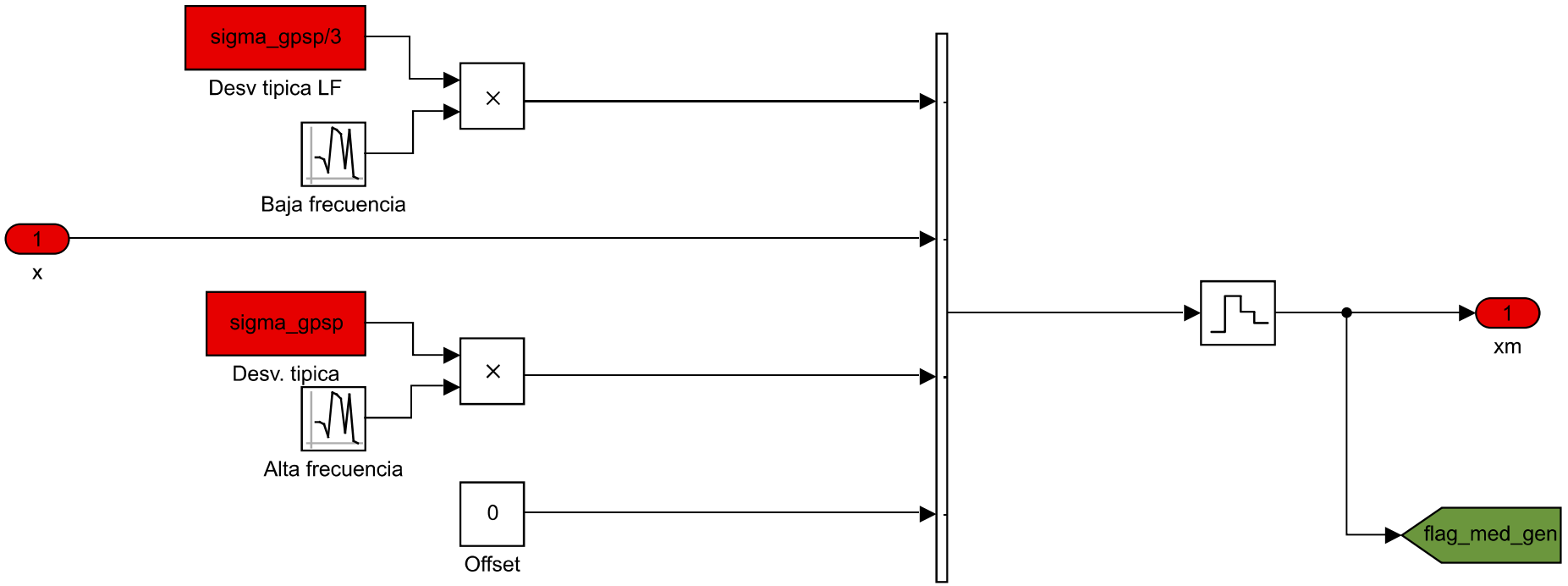

x
xm
Espacio estados
1
q3
q3
select_bar
> 0
Switch
q3_bar
1
q3bar
x
xm
HF + LF

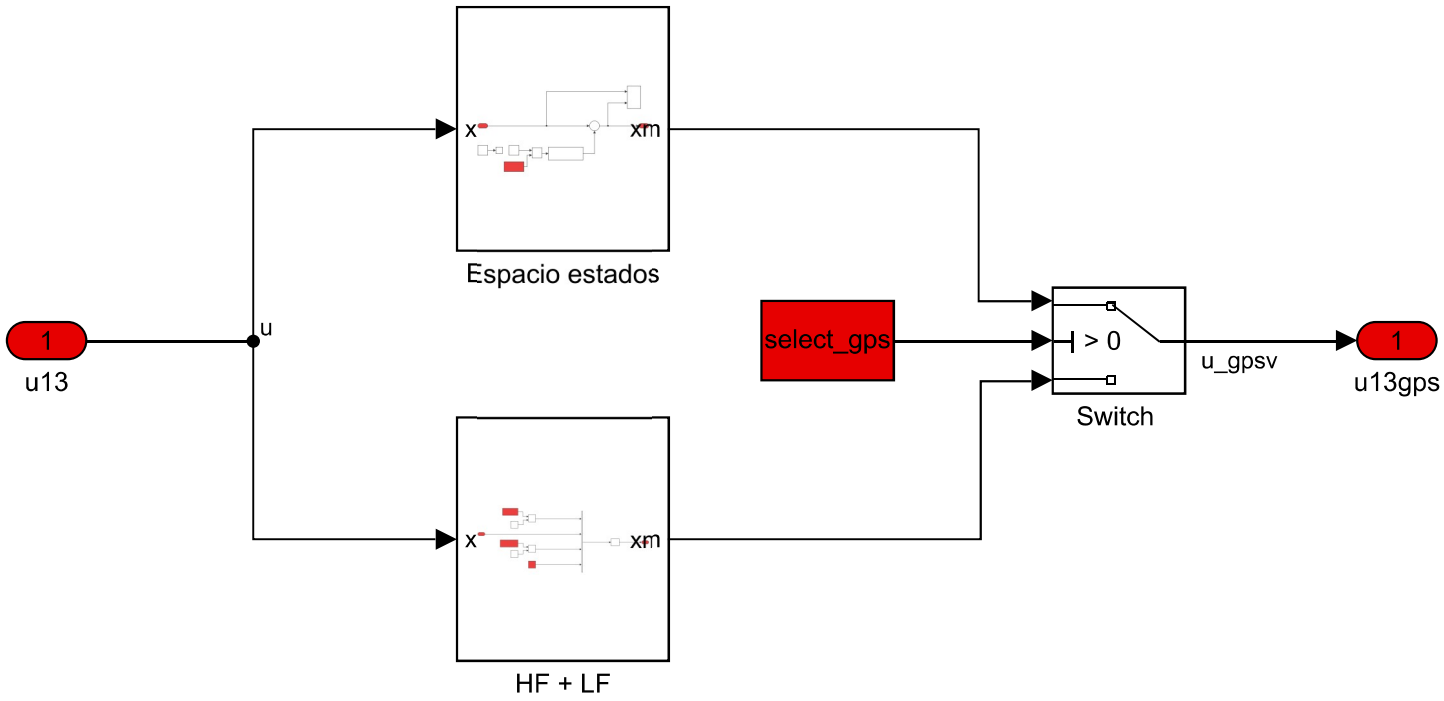

1
x
1
xm
+
+
×
$\dot{x} = Ax + Bu$
$y = Cx + Du$
sigma_bar

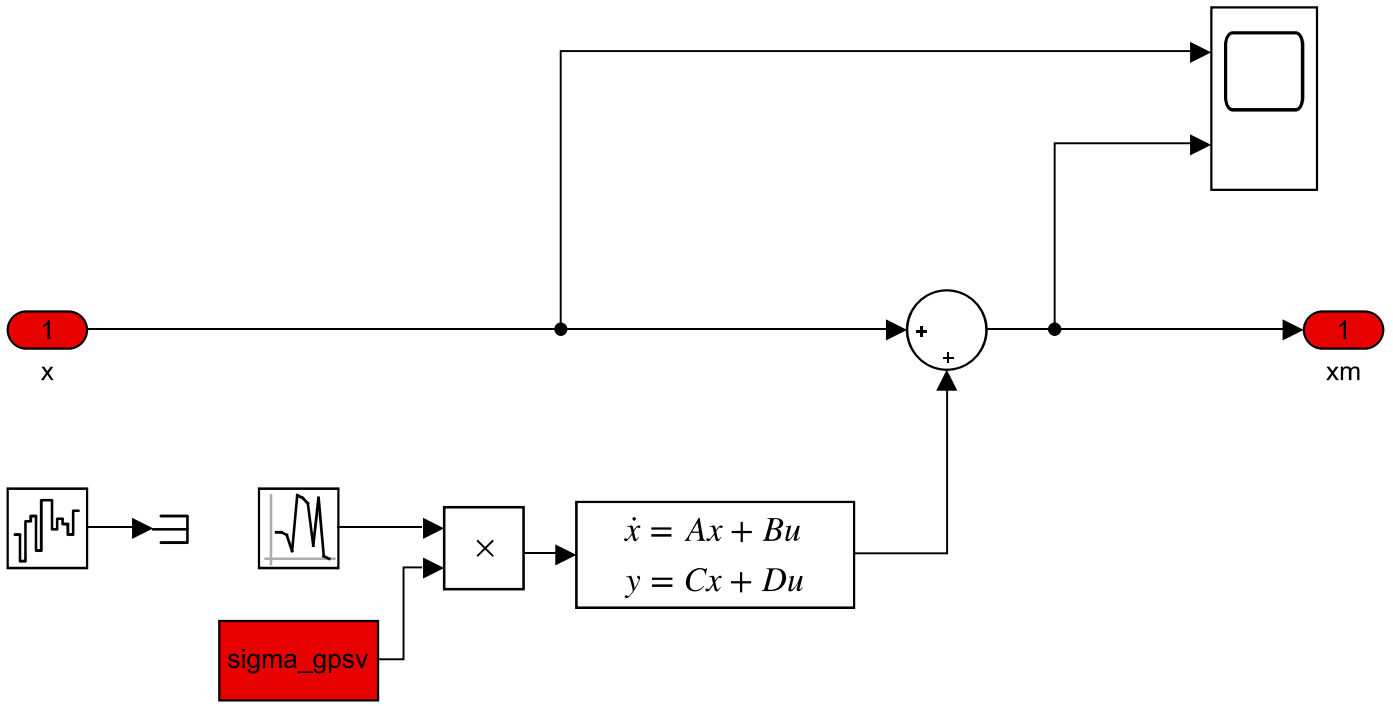

sigma_bar/3
Desv tipica LF
Baja frecuencia
1
x
sigma_bar
Desv. tipica
Alta frecuencia
0
Offset
1
xm

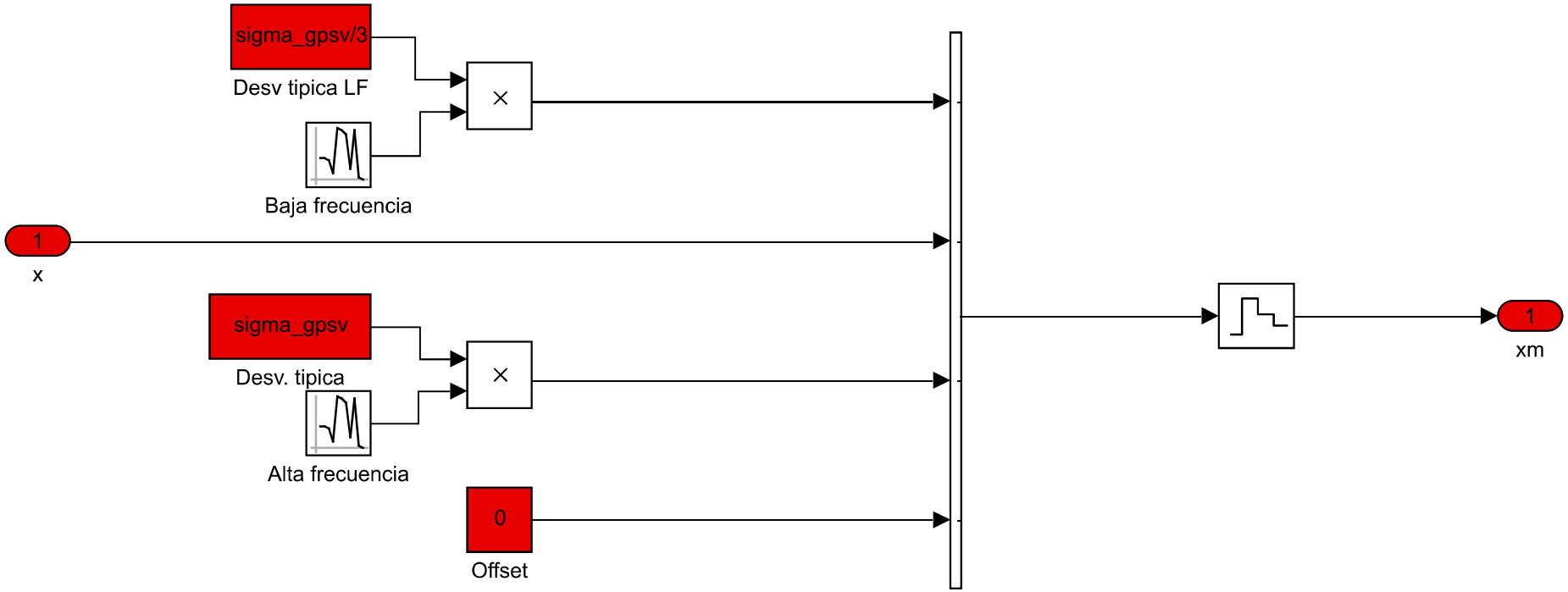

1
q13
q
x
xm
Espacio estados
x
xm
HF + LF
select_gps
> 0
Switch
q_gpsp
1
q13gps


1
x
1
xm
sigma_gpsp
×
$\dot{x} = Ax + Bu$
$y = Cx + Du$

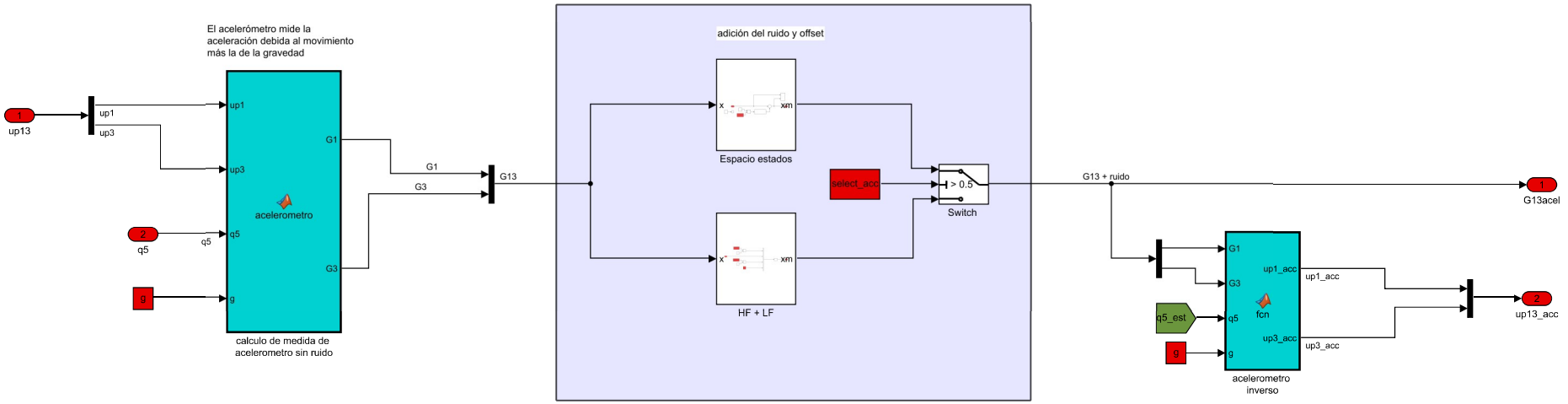

sigma_gpsp/3
Desv tipica LF
Baja frecuencia
1
x
sigma_gpsp
Desv. tipica
Alta frecuencia
0
Offset
1
xm
flag_med_gen

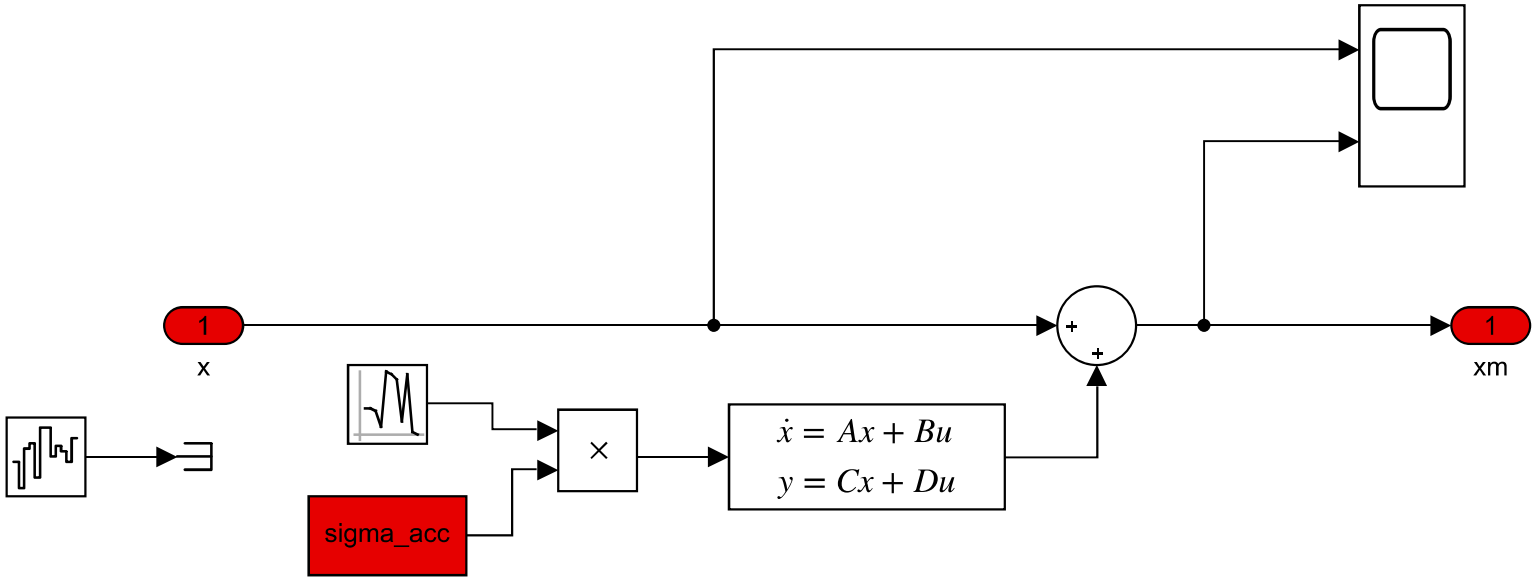

1
u13
u
x
xm
Espacio estados
x
xm
HF + LF
select_gps
> 0
Switch
u_gpsv
1
u13gps

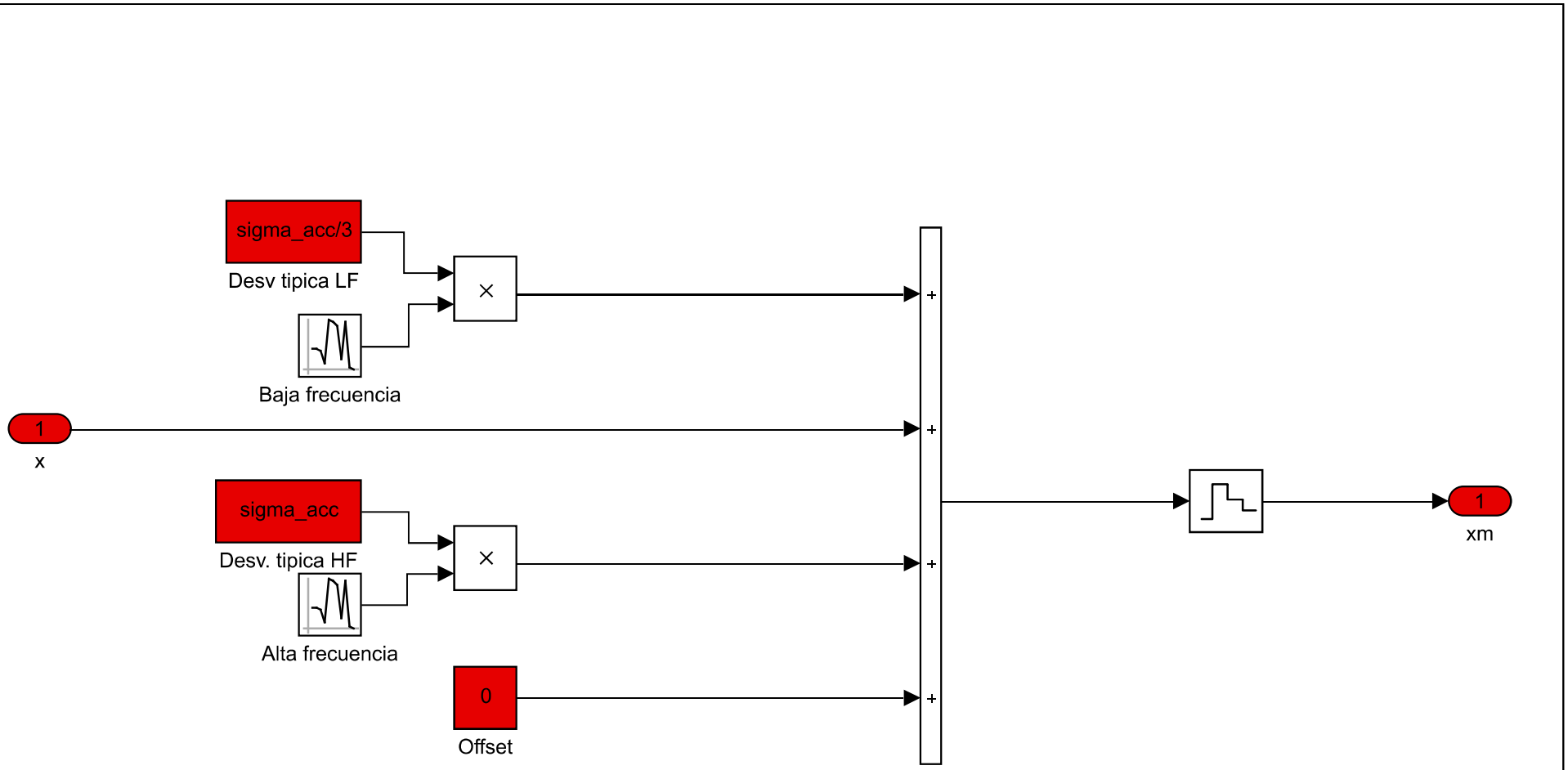

1
x
+
+
1
xm
×
$\dot{x} = Ax + Bu$
$y = Cx + Du$
sigma_gpsv



uav_con_manipulador_con_sensores_2d_Q5KF/Modelo sensores/GPSv/HF + LF	26


El acelerómetro mide la aceleración debida al movimiento más la de la gravedad
1
up13
up1
up3
up1
up3
G1
G3
acelerometro
2
q5
q5
q5
g
g
calculo de medida de acelerometro sin ruido
G1
G3
G13
adición del ruido y offset
x
xm
Espacio estados
x'
xm
HF + LF
select_acc
> 0.5
Switch
G13 + ruido
1
G13acel
G1
G3
q5_est
g
q5
g
fcn
up1_acc
up3_acc
up1_acc
up3_acc
acelerometro inverso
2
up13_acc

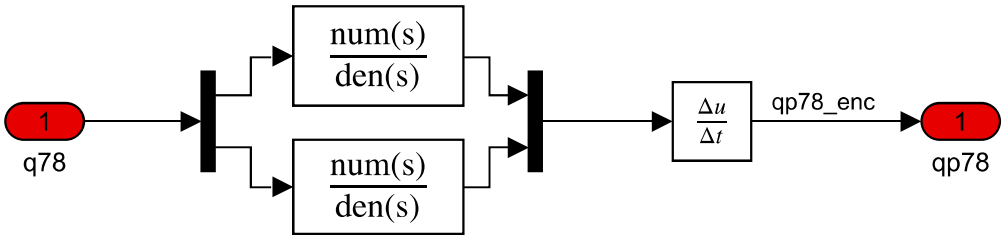

1
x
sigma_acc
×
$\dot{x} = Ax + Bu$
$y = Cx + Du$
+
+
1
xm

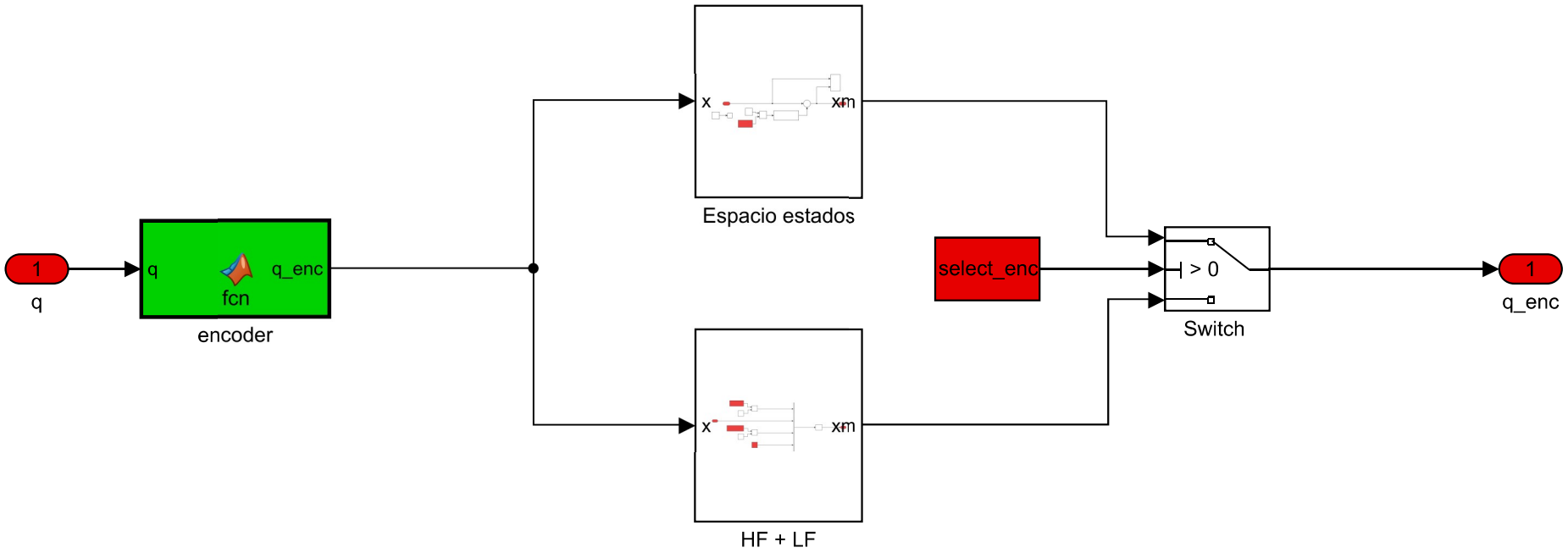

sigma_acc/3
Desv tipica LF
Baja frecuencia
1
x
sigma_acc
Desv. tipica HF
Alta frecuencia
0
Offset
1
xm

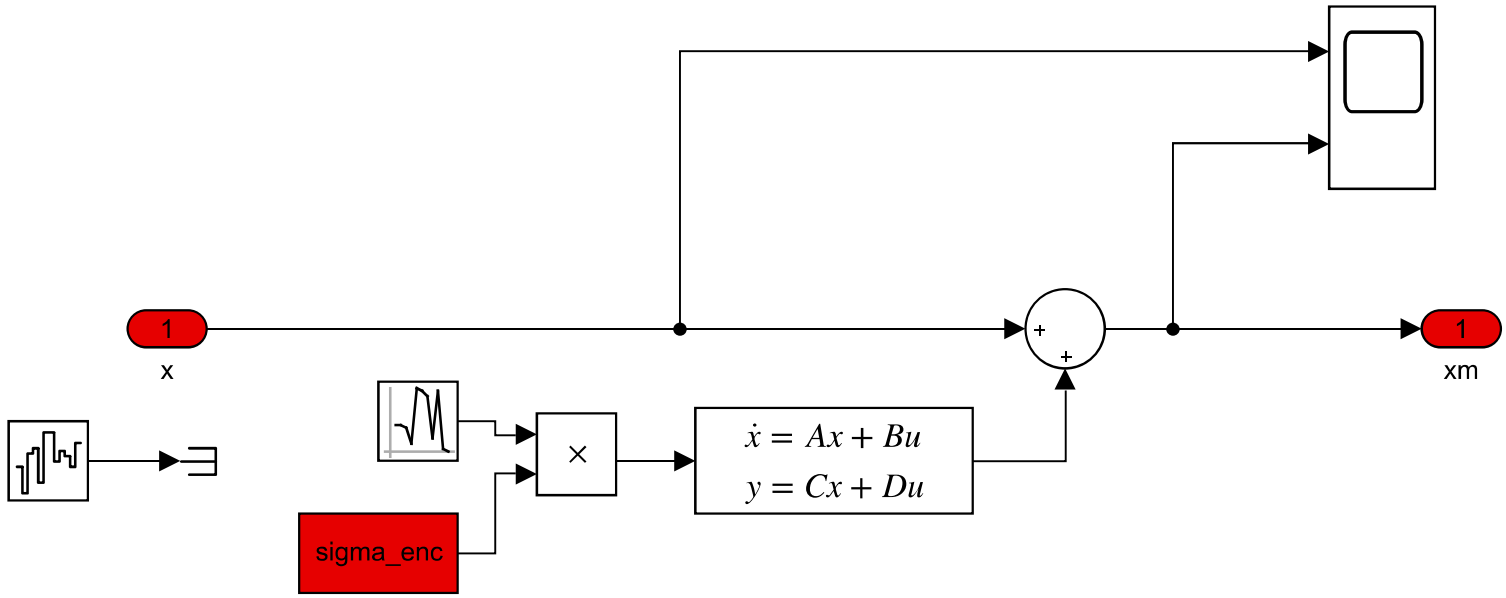
```
function [up1_acc,up3_acc] = fcn(G1,G3,q5,g)

up1_acc=cos(q5)*G1+sin(q5)*G3;
up3_acc=(-sin(q5)*G1+cos(q5)*G3)+g;
```

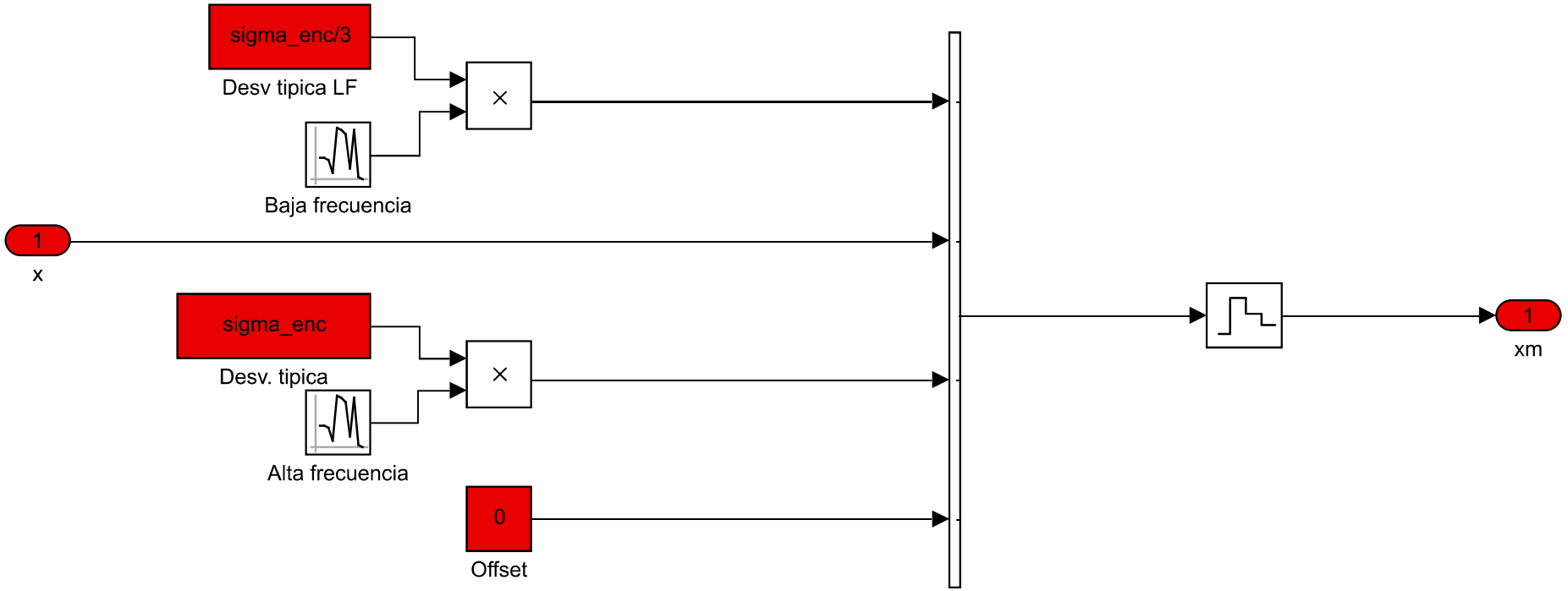
```
function [G1,G3] = acelerometro(up1,up3,q5,g)

%% Antes de cambiar el sentido de q5
% G1=-g*sin(q5)+up1*sin(q5)+up3*cos(q5);
% G3=-g*cos(q5)+up1*cos(q5)-up3*sin(q5);

%% Con el signo y sentido de q5 correcto, con la matriz de rotacion
G1=cos(q5)*up1-sin(q5)*(up3-g);
G3=sin(q5)*up1+cos(q5)*(up3-g);
```

Subsistema Derivación

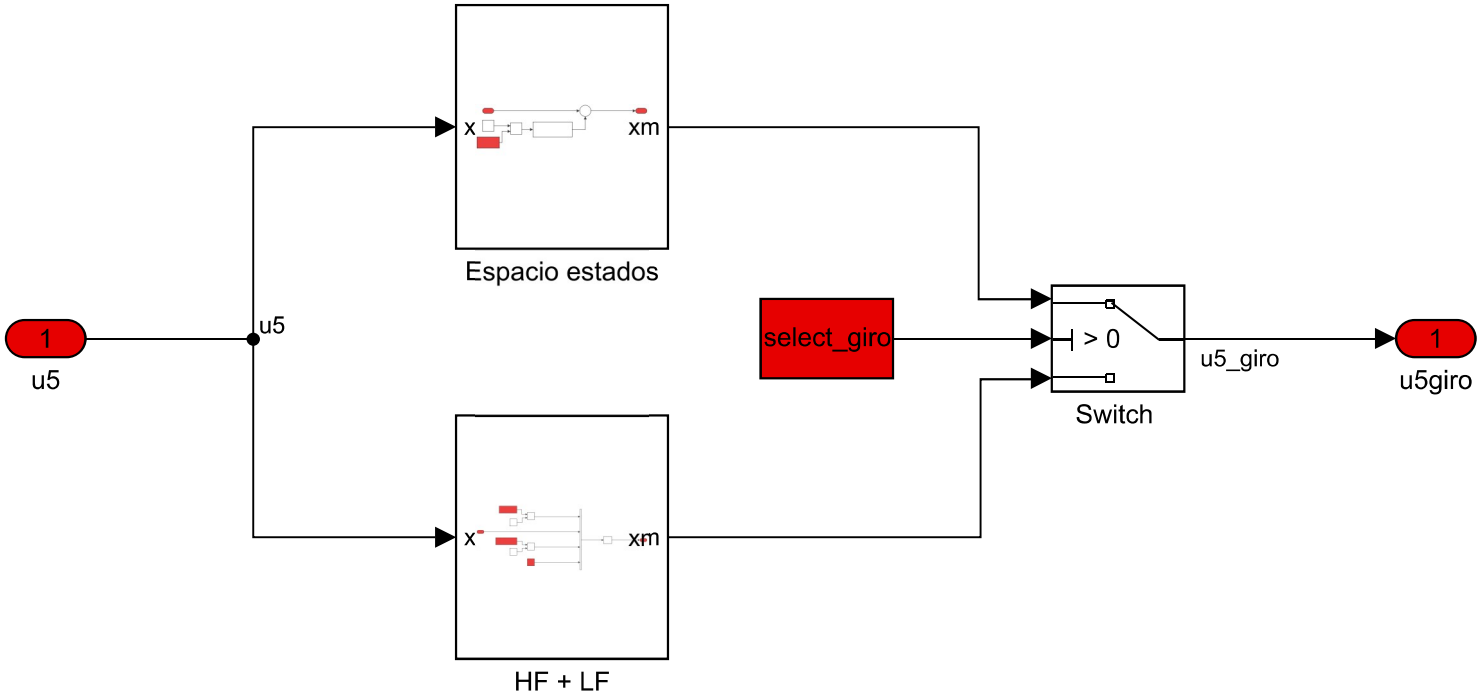


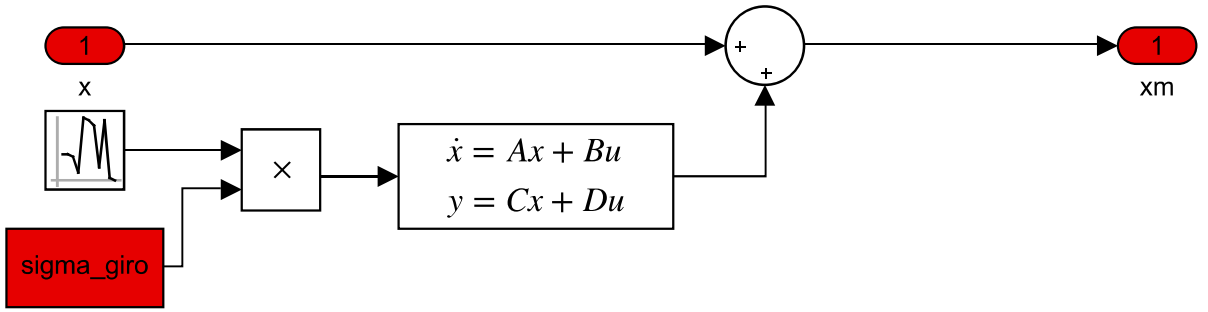

1
q
q
fcn
q_enc
encoder
x
xm
Espacio estados
x
xm
HF + LF
select_enc
> 0
Switch
1
q_enc

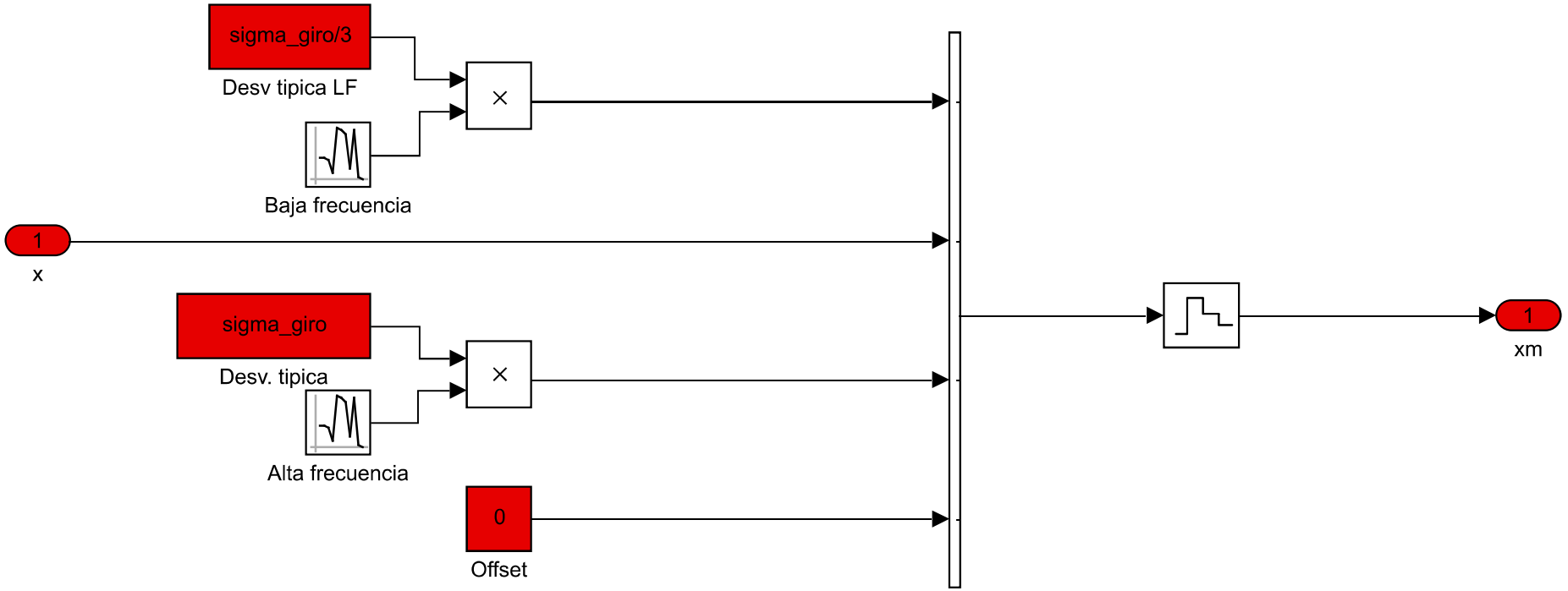

1
x
sigma_enc
×
$\dot{x} = Ax + Bu$
$y = Cx + Du$
+
+
1
xm

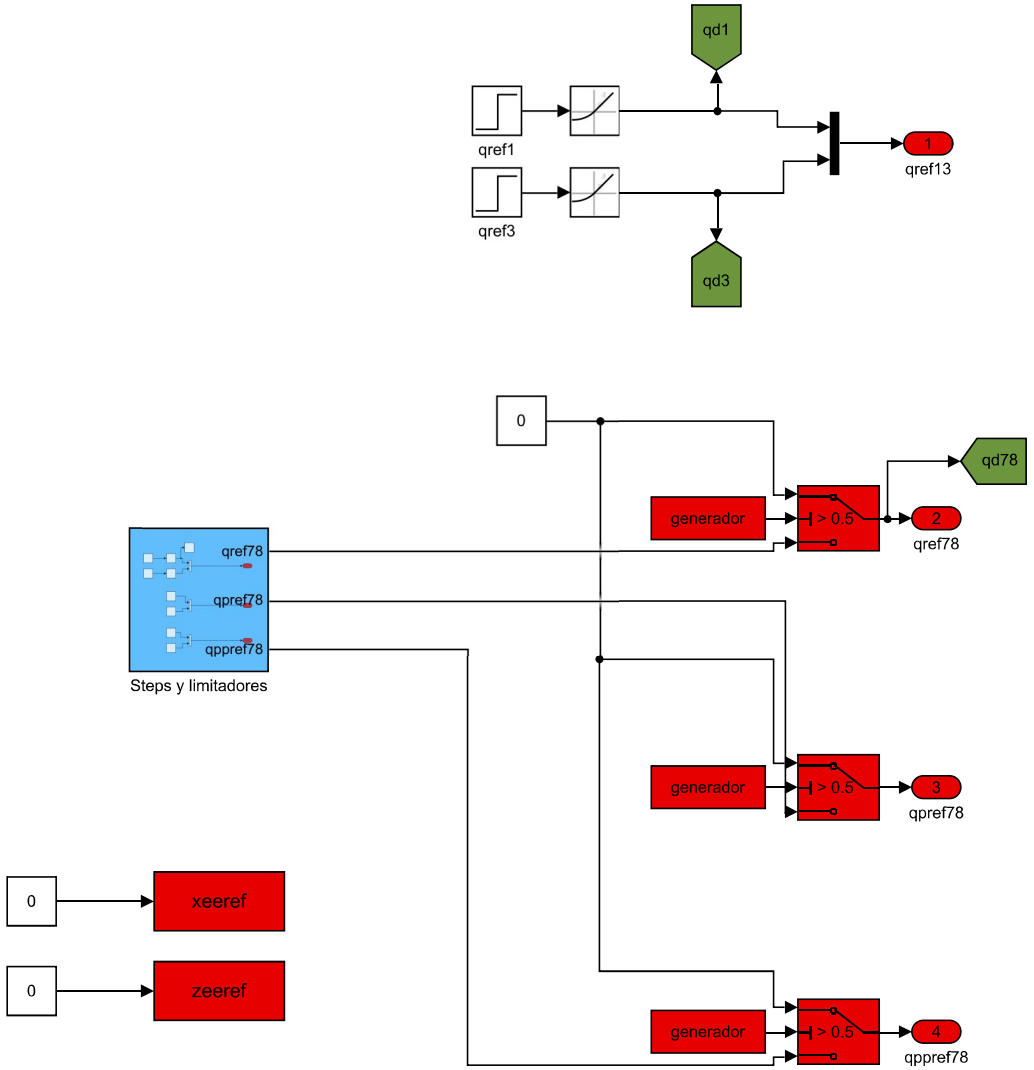

sigma_enc/3
Desv tipica LF
Baja frecuencia
1
x
sigma_enc
Desv. tipica
Alta frecuencia
0
Offset
1
xm

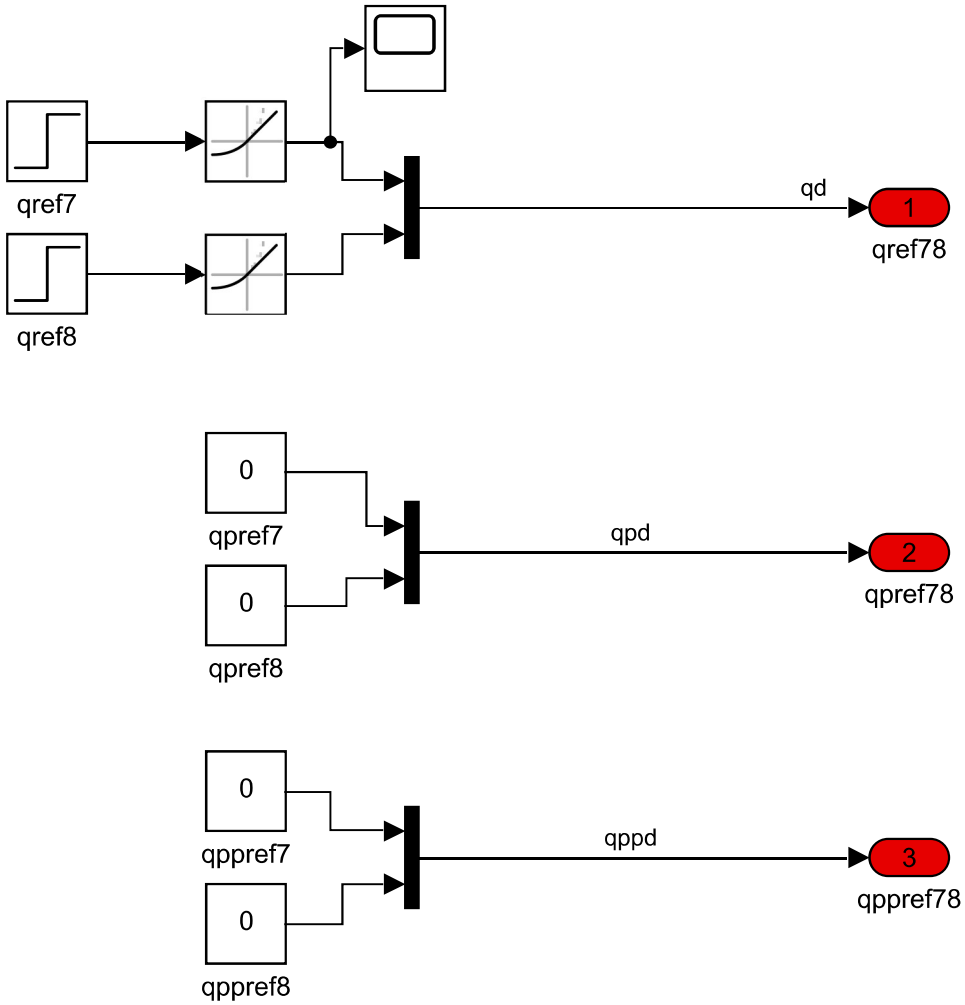
```
function q_enc = fcn(q)

res=360/1024;   %grados
res=res*pi/180; %rad

aux=round(q/res);

q_enc=aux*res;
```


1
u5
u5
x
xm
Espacio estados
x
xm
HF + LF
select_giro
> 0
Switch
u5_giro
1
u5giro


1
x
sigma_giro
×
$\dot{x} = Ax + Bu$
$y = Cx + Du$
+
+
1
xm


sigma_giro/3
Desv tipica LF
Baja frecuencia
1
x
sigma_giro
Desv. tipica
Alta frecuencia
0
Offset
1
xm


qd1
qref1
qref3
1
qref13
qd3
0
qd78
generador
> 0.5
2
qref78
qref78
qpref78
qppref78
Steps y limitadores
generador
> 0.5
3
qpref78
0
xeeref
0
zeeref
generador
> 0.5
4
qppref78



uav_con_manipulador_con_sensores_2d_Q5KF/Referencias/Steps y limitadores	41